

AfriPlaybook

A practical guide to building high-quality datasets for African languages

Table of contents:

Welcome

- [Who this playbook is for](#)
- [What you will learn](#)
- [How to read it](#)
- [Free and open](#)

Introduction

- [Scraping will not fix this](#)
- [The field is growing, the data is not keeping up](#)
- [Why we wrote this playbook](#)
- [Built in the open](#)
- [How to cite this playbook](#)

How to read this playbook

How to Contribute

- [Ways to contribute](#)
- [Pick the path that fits your change](#)
- [Path 1: Edit on the site \(easiest\)](#)
- [Path 2: Edit one file on GitHub](#)
- [Path 3: Fork and pull request \(for chapters and larger changes\)](#)
 - [Step 1: Open an issue first](#)
 - [Step 2: Fork and clone](#)
 - [Step 3: Install and run locally](#)
 - [Step 4: Create a branch](#)
 - [Step 5: Add or edit your content](#)
 - [Step 6: The sidebar updates itself](#)
 - [Step 7: Preview your change](#)
 - [Step 8: Run the build](#)
 - [Step 9: Commit and push](#)
 - [Step 10: Open a pull request](#)
- [Writing guidelines](#)
- [Translating a page](#)
- [Get help](#)

Our contributors

- [Not listed here?](#)

How to Write a Chapter

- [Writing in Markdown](#)
 - [Sections and subsections](#)
 - [Admonitions: the colored callout boxes](#)
 - [When to use which box](#)
 - [Other things that make a page nicer](#)
- [Writing in Microsoft Word](#)
 - [Headings map one level down](#)
 - [Admonitions in Word](#)
 - [A few more Word tips](#)
- [Preview before you submit](#)

Core principles

- [Build with speakers, not just for them](#)
- [Consent and ownership come first](#)

- Quality over scale
- Context travels with the data
- Build in the open, design for reuse
- Do no harm

What this playbook is (and isn't)

- The one-sentence framing
- What this playbook is not
- What this playbook is
- Why this scope: the evidence
- What comes next: the phased plan
 - Phase 1: the two highest-leverage additions
 - Phase 2: cross-cutting realism
 - Phase 3: durable governance
 - Phase 4: compute-poor practice
 - Phase 5: finding current resources (redesigned)
- What we deprioritise
- The one rule that keeps this playbook honest

Before You Start

- Step 1: Search for prior work
- Step 2: Judge what you find
- Step 3: Reuse, extend, or build
- Step 4: Learn from the people who built it
- Worked examples

Named Entity Recognition

- Step 1: What a search turns up
- Step 2: Judging MasakhaNER 2 against the checklist
- Step 3: Reuse, extend, or build
- Step 4: Learning from the people who built it

Sentiment analysis

- Step 1: What a search turns up
- Step 2: Judging AfriSenti against the checklist
- Step 3: Reuse, extend, or build
- Step 4: Learning from the people who built it
- Further reading

Hate speech and content safety

- Step 1: What a search turns up
- Step 2: Judging AfriHate against the checklist
- Step 3: Reuse, extend, or build
- Step 4: Learning from the people who built it
- Further reading

Dataset search log

- Project
- Step 1: Searches
- Step 2: Datasets found
- Step 3: Decision
- Step 4: People contacted

Project Management

- 1. Scope the project
- 2. Settle ethics, consent, and the law early

- 3. Plan the work
- 4. Budget and fund the work
- 5. Build the team
- 6. Coordinate and track
- 7. Manage risks
- 8. Release and hand over
- Checklist before you start

Project charter template

- How to use this template
- Project charter: [PROJECT NAME]
- 1. Purpose
- 2. The parties
- 3. Language and orthography
- 4. Data: what is collected, from whom, how
- 5. Community IP and licence
- 6. Team, timeline, and budget
- 7. Governance and decision-making
- 8. Deliverables and commitments
- 9. Ethics and safeguards
- 10. Signature block
- 11. Amendments

Data Governance

- Govern for the community, not just the dataset
 - Community ownership and self-determination
 - FAIR and CARE, together
- Consent and rights
 - Informed consent
 - Rights over sourced and contributed data
- Licensing
 - Data and code are licensed differently
 - African and community licences
 - Attribution and reuse
- Privacy and sensitive data
 - Personally identifiable information
 - Sensitive and restricted content
 - The law that applies: African data-protection regimes
- Ethics and harm avoidance
 - A lightweight ethics review
 - Bias, representation, and harm
- Community ownership and benefit-sharing
 - Governance models
 - Benefit-sharing and credit

Governance Toolkit

- What is in the toolkit
- How the pieces fit
- What the toolkit is not

Licensing decision tree

- Work through the tree
- After the tree

Contributor Agreement Template

- [When to use this template](#)
- [What this agreement covers](#)

Annotator contract template

Sustainability Plan Template

- [When to use this template](#)
- [What this plan covers](#)

Data Collection

Data Modalities

- [What a modality is](#)
- [The modalities you'll most often collect](#)
 - [Text](#)
 - [Speech and audio](#)
 - [Images](#)
 - [Video](#)
 - [Multimodal data](#)
- [Why this decision comes first](#)

Data Sources

- [Why "source" and "method" are different questions](#)
- [The main categories](#)
 - [The open web](#)
 - [Institutional and government archives](#)
 - [Community and oral sources](#)
 - [Broadcast and media archives](#)
 - [Existing datasets and corpora](#)
 - [Religious and literary texts](#)
 - [Purpose-built collection campaigns](#)
- [Weighing sources against each other](#)

Web Scraping

- [What scraping is, and what it isn't](#)
- [The access landscape keeps shifting](#)
- [A framework for deciding whether and how to scrape](#)
- [Practical guidance](#)
- [A minimal, well-behaved scraper](#)
- [When scraping is the right call](#)

Application Programming Interfaces (APIs)

- [How this differs from scraping](#)
- [A cautionary, very real example](#)
- [What APIs are well suited for](#)
- [Practical guidance](#)
- [A worked example: pulling from a Wikimedia API](#)
- [The bigger pattern](#)

Case Study: Multimodal Data Collection

- [The task](#)
- [Decision 1: the image layer was reused, not collected fresh](#)
- [Decision 2: cross-cultural comparison sets a constraint before annotation starts](#)
- [Decision 3: cost and labor planning had to scale by language, not just by image](#)
- [Takeaways for your own multimodal project](#)

Cost and Resource Planning

- [Why Planning Matters](#)
- [Key Components of Planning](#)
 - [Budgeting Annotation Costs](#)
 - [Time Estimation](#)
 - [Scaling Strategy](#)

[Data Cleaning and Preprocessing](#)

- [Why Data Cleaning Matters](#)
- [Key Steps in Data Cleaning](#)
 - [Deduplication, Normalization, and Filtering](#)
 - [Language Detection and Formatting](#)
 - [Noise and Toxicity Handling](#)
 - [Missing and Corrupted Data Handling](#)

- [A cleaning pipeline, end to end](#)

- [A note on the cleaned format](#)

[Data Provenance and Traceability](#)

- [Why Provenance Matters](#)
- [Key Components of Provenance](#)
 - [Source Tracking](#)
 - [Data Lineage](#)
 - [Transformation Logs](#)
- [A dataset card you can version](#)
- [A transformation log that travels with the data](#)

[Ethics, Bias, and Governance](#)

- [Why Ethics and Governance Matter](#)
- [Key Components of Ethical Data Practices](#)
 - [Bias Identification and Mitigation](#)
 - [PII Detection and Removal](#)
 - [Anonymization Strategies](#)
 - [Sensitive Attribute and Content Handling](#)
 - [Fair Representation](#)
 - [Risk Documentation and Transparency](#)

[Annotation Task Design and Human Factors](#)

- [Estimate task complexity before you scale](#)
- [Design against annotator fatigue](#)
- [Let the interface do some of the work](#)
- [A labeling configuration](#)
- [From a config to a running project](#)

[Training and Guidelines](#)

- [Write guidelines around examples, not definitions](#)
- [Train and calibrate until annotators converge](#)
- [Pilot, then iterate the design](#)
- [Prove the design with a minimum viable dataset](#)

[Workflow and Adjudication](#)

- [Use multiple annotators per item](#)
- [Assign work with deliberate redundancy](#)
- [Resolve disagreement, but read it first](#)

[Inclusive and Bias-Aware Annotation](#)

- [Why annotator background shapes the data](#)
- [Recruit for diversity, deliberately](#)

- Record annotator context, with consent
- Build awareness, and involve the community

Annotation guidelines template

- How to use this template
- [PROJECT NAME] annotation guidelines
- 1. What we are annotating and why
- 2. The team
- 3. Definitions
- 4. The annotation itself: decision procedure
 - 4.a Decision tree
 - 4.b Corner cases
- 5. Label set
- 6. Orthography, script, and diacritics
- 7. Inter-annotator agreement
- 8. Adjudication
- 9. Change log
- 10. Contact and community

Data Quality Assurance

- Measure inter-annotator agreement
- Use gold data and control checks
- Audit and spot-check by hand
- Close the feedback loop

Data Quality Management

- Handle class imbalance
- Detect noise and outliers
- Build an error-analysis pipeline
- Version the dataset

Community Ecosystems

- The communities and what they do
- Academic and industry collaboration
- Contributing, and crediting contributors

Collaboration and Shared Tasks

- Shared tasks and benchmarks
- Workshops and open challenges

Text

- Shared across text data
- Tasks

Defining text classification tasks:

- Task distinction

Collecting & Preparing Data

- Data sources
- Collection & selection approaches
 - Collection strategy guide
 - Comparisons of data selection methods
- Processing & sampling
 - Cleaning pipeline example
- Quality control
 - Quality-control policy

Annotation

- [Choosing annotators](#)
- [Annotator safety & well-being](#)
- [Annotation tools](#)
 - [Crowdsourcing platforms](#)
 - [In-house \(self-hosted\) tools](#)
 - [Lightweight tools for small datasets](#)
 - [A labeling configuration](#)
- [Inter-annotator agreement](#)
 - [Why agreement matters](#)
 - [Percentage agreement](#)
 - [Agreement between two annotators](#)
 - [Agreement among three or more annotators](#)
 - [Deciding the final labels](#)
 - [Interpreting agreement scores](#)
 - [What to report](#)
- [Quality control](#)
 - [Pilot annotation](#)
 - [Control \(gold-standard\) questions](#)
 - [Determining the number of annotators](#)

[Sentiment Analysis](#)

- [Types of sentiment analysis](#)
- [Sentiment analysis labels definition](#)

[Emotion Analysis](#)

- [Single-label Emotion Analysis](#)
- [Multi-label Emotion Analysis](#)
- [Conditions for Determining the Final Emotion Label](#)
 - [Example 1](#)
 - [Example 2](#)
 - [Example 3](#)
 - [Example 4](#)
 - [Example 5](#)
- [Emotion Intensity](#)

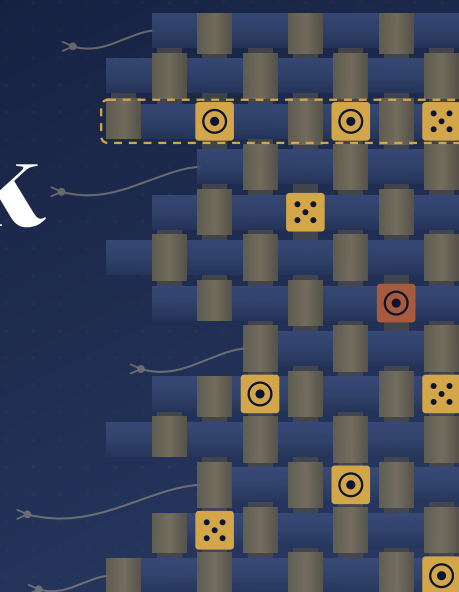
[Hate Speech Analysis](#)

[Text Generation](#)

Welcome

AfriPlaybook

A practical guide to building
high-quality datasets for
African languages



This is the website of the **AfriPlaybook**, an open guide to building datasets for African languages. It follows a dataset from first idea to public release: deciding what to collect and from whom, designing the annotation task, recruiting and paying the people who do it, checking their work, documenting the result, and publishing it so that others can build on it.

Africa is home to roughly a third of the world's languages, yet most of them have almost none of the text and speech that language technology depends on. The reason is that building good data is expensive, careful work, and few people are ever taught how to do it. Most guidance on dataset creation assumes English, a comfortable budget, and a task someone has solved before. This playbook assumes instead the conditions most African-language projects face: modest funding, volunteer or part-time teams, several languages and scripts at once, and communities who should remain the owners of what they help to create.

Who this playbook is for

The playbook is written for anyone who builds, or wants to build, a dataset for an African language, and for NLP researchers working on any language where data is scarce. It draws its examples from African projects, but the problems it deals with are not unique to Africa: little usable text online, several scripts or dialects within one language, small budgets, and communities whose consent and ownership must be respected. Most of its advice applies wherever those conditions hold. Among its readers we expect students, researchers, linguists and language activists who want their language to be usable by machines, community organisers running a collection drive, and engineers who have discovered that the data they need does not exist. No background in machine learning is assumed. What is assumed is a language you care about and a willingness to do careful work.

What you will learn

The chapters are grouped into parts:

- **Foundations** covers the work common to every project: planning, collecting data, designing annotation, governing data, assuring its quality, and working with communities.
- **Text** covers text classification, text generation and machine translation.
- **Speech** covers speech recognition, text-to-speech, speech translation, audio understanding, emotion recognition and speaker diarization.
- **Vision** covers image data, document AI and OCR, and sign language and video.
- **Multimodal** covers tasks that pair images with text, and the use of large language models to help build data.
- **Lifecycle & Release** covers evaluation, documentation, release, deployment, cross-language transfer, and the law and ethics of consent.
- **Templates** gives you documents to adapt, such as consent forms, annotation guidelines and dataset cards.
- **Case Studies** describes real projects, written by the people who ran them.

Each of the task parts explains what changes when you build data for that kind of task.

How to read it

You do not have to read the playbook in order. If you are starting a project from nothing, begin with the [Introduction](#), which explains why African languages lack data and why scraping the web will not fix it, and then read the Foundations chapters in sequence. If you already know your task, go to [Before You Start](#), which lists existing resources for common tasks and helps you decide whether to extend an existing dataset or build a new one. [How to read this playbook](#) suggests other routes through the book, and the [glossary](#) defines its terms. If you work offline or on a slow connection, the whole playbook is available as a single PDF from the **AfriPlaybook** menu at the top of the page.

Free and open

This website is free to read and will remain so. The playbook is maintained by the Waraka community, Masakhane, and AfricaNLP researchers, with support from the [Masakhane African Languages Hub](#)[↗]. It improves only as far as its readers improve it. If you find an error, have run a project whose lessons others should hear, or can translate a page, please [contribute](#). The site interface is available in Hausa, Amharic, Swahili, French and Portuguese, and translations of the chapters are added as volunteers complete them. Questions and disagreements are welcome in [GitHub Discussions](#)[↗] and on [Discord](#)[↗].

If the playbook informs your research or teaching, please [cite it](#).

 [Cite this page](#)  4 min read

CONTRIBUTORS  (<https://github.com/shmuhammad>)  (<https://github.com/abumafri>)  (<https://github.com/seyyaw>)

Introduction

💡 CONTRIBUTE

This playbook is open source and community-owned. You don't need to write a whole chapter to help. Fixing an error, translating a page, or sharing what worked on a real project all count. See [Contribute](#).

The bullet was the means of the physical subjugation. Language was the means of the spiritual subjugation.

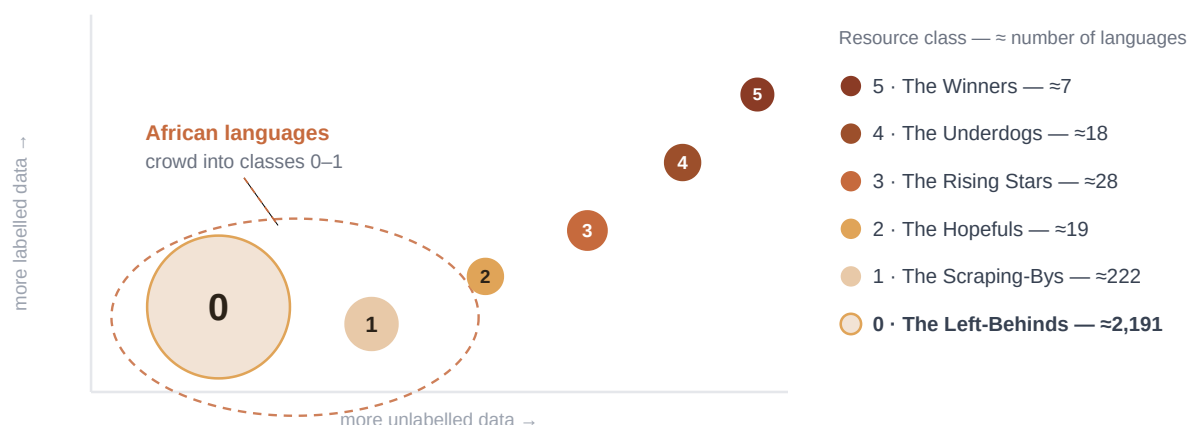
Ngũgĩ wa Thiong'o, Decolonising the Mind: The Politics of Language in African Literature (1986)

Africa is home to almost a third of the world's living languages: about 2,140 of the roughly 7,160 spoken on Earth ([Ethnologue, 2024](#)). Most of them remain out of reach of the systems now reshaping how the rest of the world reads, writes, searches, translates, and speaks. Even the best-supported, such as Swahili, Hausa and Amharic, lag far behind high-resource languages. When a language model mangles Yorùbá, Chichewa, or Wolof, the cause is usually not the model but the data. The text and speech these systems learn from barely exist in a usable form.

[Joshi et al. \(2020\)](#) sort the world's languages into six tiers by how many resources they have. The bottom tier, the *left-behinds*, with essentially no labelled data and little prospect of being served by current methods, holds the overwhelming majority of languages, and African languages crowd into it. Most have no annotated corpus, no benchmark, no tools. Many have tens of millions of speakers. What they lack is data, because almost no one has built it. The picture still holds, though the field now treats *low-resource* as multidimensional: a matter of tools, speakers, funding, and institutional support as much as raw data, with no single agreed definition ([Ranathunga & de Silva, 2022](#); [Nigatu et al., 2024](#)).

Most of the world's languages have almost no data

The six resource classes of Joshi et al. (2020), by data available



Adapted from Joshi et al. (2020). Bubble size and counts are approximate; darker shading marks better-resourced classes.

Scraping will not fix this

When a language has no data, the instinct is to go and scrape more of it: crawl a wider slice of the web and trust that coverage will follow. For African languages, that instinct fails.

The web does not contain much African-language text, and what it contains is thin and noisy. When [Kreutzer et al. \(2022\)](#) audited the large multilingual crawls everyone trains on, they found that for many low-resource languages a large share of the data was mislabelled, machine-translated, or not language at all. At the tail, quality collapses along with quantity.

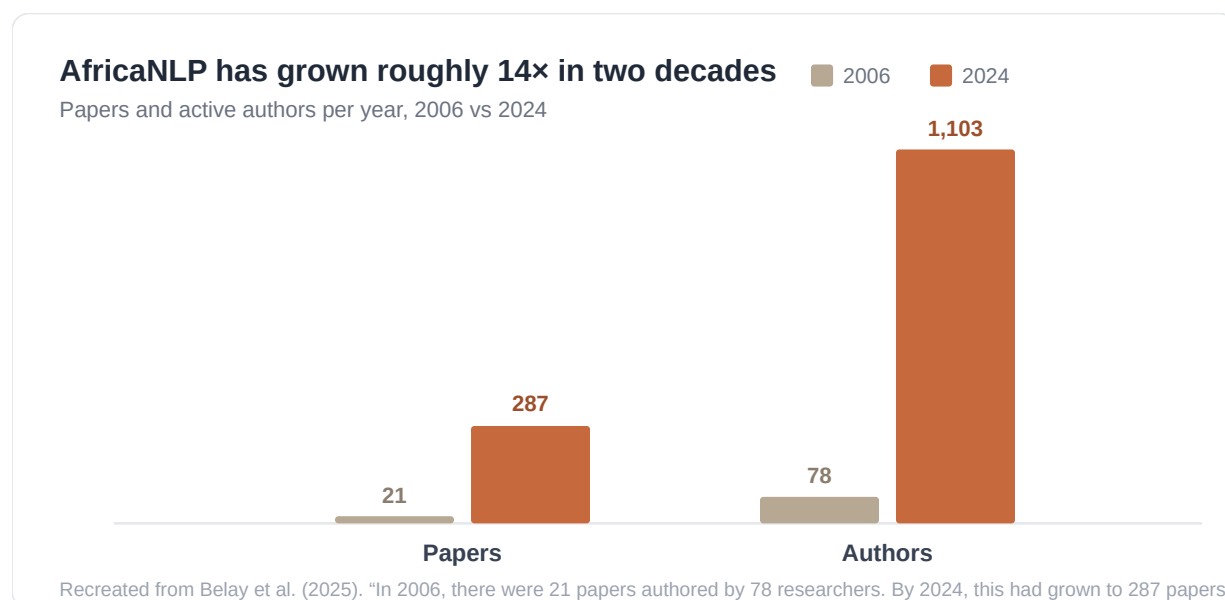
The only sure way to get high-quality data for African languages is to build it with the people who speak them, the people who know the words, the grammar, the idioms, and the culture. One of the main obstacles in AfricaNLP is that the people who speak these languages are not the ones building the data. The people who build it often cannot tell what is correct, what is offensive, or what is missing. They do not know what matters to the communities behind the language, or how to keep the data they collect from causing harm.

That gap has real consequences. Data built without its speakers can look clean while being quietly wrong, and any model trained on it inherits every mistake. Such errors spread into search results, translations, and the everyday tools that millions of people are starting to depend on. Getting the data right decides whether a language is served well, served badly, or left out of these tools altogether.

This playbook is about how to fix that problem. It is a practical, opinionated, step-by-step guide to building high-quality datasets for African languages, drawing on the direct experience of the people who speak and understand them. The playbook is built by the people who know the languages, for the people who want to build datasets for them. It is about how to do it right, and how to do it safely.

The field is growing, the data is not keeping up

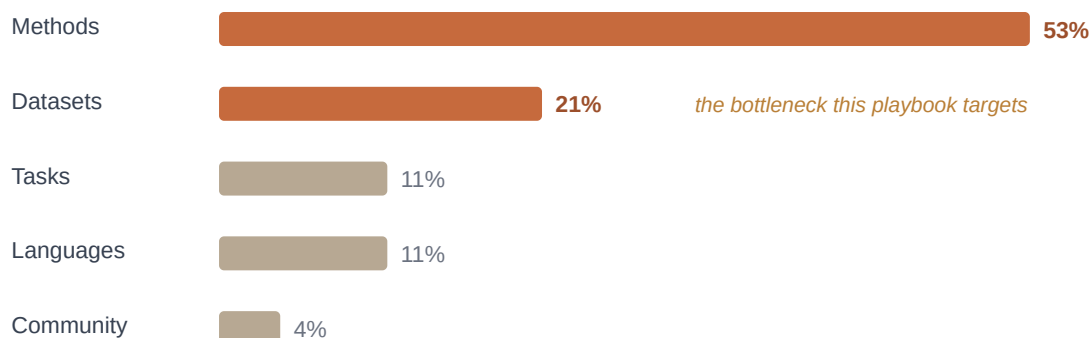
Over the past two decades, AfricaNLP has grown from a niche interest into an established field. Research output has grown more than tenfold, from about 20 papers a year in 2006 to nearly 300 in 2024 ([Belay et al., 2025](#)).



But more papers have not meant more data. More than half of this work proposes new methods, while only about one in five contributions introduces a new dataset ([Belay et al., 2025](#)). We are getting better at building models faster than we are building the data they learn from.

Where AfricaNLP effort has gone, 2005–2025

Share of contribution sentences, by what the work produced



Recreated from Belay et al. (2025), Table 4. Method and dataset shares aggregate knowledge- and artifact-type sentences.

Methods and datasets are not made the same way. A method can often be reused across languages; a dataset has to be built for each one, from scratch, by people who speak it. That means recruiting annotators, writing guidelines, running quality control, and securing consent. It is slow, unglamorous work, and it is rarely funded, least of all for African languages.

This playbook exists to make that work easier. It walks through each stage of building a dataset: deciding what to collect, designing the annotation, checking quality, documenting, and releasing. It is written for the real conditions of African-language NLP: limited data, multilingual teams, scarce funding, and communities who should remain the owners of what they help create.

Why we wrote this playbook

Almost every guide to building datasets quietly assumes English, a generous budget, and a problem someone has already solved once. Little of that holds when you are starting a corpus for a language with no prior resources, a volunteer team, and decisions to make that the literature never covers.

The AfriPlaybook is the manual we wish we had had. Many of us learned to build datasets for African languages the hard way, through trial and error, with little written down and few people to ask. That is part of why datasets have lagged so far behind methods. This playbook collects what we learned so that the next team can start sooner, avoid the mistakes we made, and build data the world can trust and reuse.

The playbook will keep growing as the community shares what it learns. The more we pool our experience, the cheaper it becomes to build data, and the sooner datasets can catch up with methods.

Built in the open

This playbook is open source, maintained by the Waraka community, Masakhane, and AfricaNLP researchers. It is not just for researchers. It is for everyone who builds datasets for African languages: volunteers, students, community organizers, and professionals alike. The people who build datasets know best what a guide like this should say, so it is only as good as the people who contribute to it. There are many ways to help:

- **Write** a chapter or section that fills a gap.

- **Review** existing chapters: correct an error, sharpen a claim, add a reference.
- **Share a case study** from a real project, including what went wrong.
- **Open a discussion** when you disagree with an approach. Disagreement makes the guide better.

Start with the [contribution guide](#), raise an idea in [GitHub Discussions](#), or join us on [Discord](#). If you build datasets for African languages, or want to learn how, you are already part of who this is for. Come and build it with us.

How to cite this playbook

If the AfriPlaybook informs your research, teaching, or project, please cite it.

BibTeX:

```
@misc{waraka2026playbook,  
  author    = {{Waraka Community}},  
  title     = {AfriPlaybook: A Practical Guide to Building High-Quality Datasets for African Languages},  
  year      = {2026},  
  publisher = {Waraka Community},  
  url       = {https://afriplaybook.waraka.org/},  
  note      = {Open-source community resource}  
}
```

Plain text (APA-style):

Waraka Community. (2026). *AfriPlaybook: A Practical Guide to Building High-Quality Datasets for African Languages*.
<https://afriplaybook.waraka.org/>

For other formats (MLA, Chicago, etc.) and a machine-readable [CITATION.cff](#), see the [/cite](#) page.

If you reference a specific chapter, please include the chapter title and its URL.

 [Cite this page](#) ⌚ 6 min read

CONTRIBUTOR  (<https://github.com/shmuhamadd>)

How to read this playbook

The playbook follows a dataset from start to finish. The map below groups its chapters into the four phases of building a dataset. Select any chapter to go straight to it.

1 START HERE

Introduction

Why African-language data matters, and how this guide helps.

2 BUILD THE DATASET

Data Collection

Plan scope, sources, cost, ethics, and provenance.

Annotation Design & Workforce

Design tasks, write guidelines, and manage annotators.

Modality-Specific Task Design

Text, speech, vision, and multimodal data.

LLM-Assisted & Synthetic Data

Use models to help build data, with safeguards.

3 MAKE IT TRUSTWORTHY & RELEASE

Data Quality & Validation

Agreement, review, and quality control.

Documentation & Governance

Consent, licensing, ownership, and datasheets.

Lifecycle & Release

Release, host, identify, and steward the data for the long term.

4 USE & GROW

Evaluation

Check data integrity and get a modelling starter kit.

Community & Collaboration

Work together and sustain the ecosystem.

Worked example: Text Classification

An end-to-end task, from raw data to labels.

You do not have to read it in order. Pick the route that fits where you are:

- **You are new to building datasets.** Read the [Introduction](#), then work through the Foundations chapters in order: [Project Management](#), [Data Collection](#), [Data Governance](#), [Annotation Design](#) and [Data Quality](#). Each builds on the one before.
- **You know your task and want to avoid repeating work.** Start with [Before You Start](#). It shows you how to find existing datasets, judge whether they fit, and decide whether to reuse, extend, or build your own.
- **You already have raw data and need it annotated.** Go to [Annotation Design](#), then [Data Quality](#).
- **You are working with speech, images, video or documents.** Go to the part for your modality: [Speech](#), [Vision](#) or [Multimodal](#). Each explains what changes once you move beyond text.
- **Your language has little or no digital text.** Read [Long-tail language onboarding](#) before anything else. It covers the orthography, community and ownership decisions that come first.

- **You are using LLMs to generate or augment data.** Read [LLM-Assisted Data](#) for the trade-offs and safeguards.
- **You are preparing a dataset for release.** Read [Documentation](#), [Dataset Lifecycle](#) and [Legal, consent, and community IP](#).
- **You are offline or on a slow connection.** Download the whole playbook as one PDF from the **AfriPlaybook** menu in the navbar. It is rebuilt with every release.
- **You would rather read in another language.** Use the language switcher at the top right. Translations are maintained by the community and grow over time.

Throughout the book you will find templates you can adapt, such as consent forms, annotation guidelines and governance checklists, along with worked examples from real African-language projects and links to datasets and tools you can reuse. New terms are defined in the [glossary](#).

 [Cite this page](#)  3 min read

CONTRIBUTORS  (<https://github.com/abumafri>)  (<https://github.com/shmuhammadd>)

How to Contribute

You do not need to be an NLP researcher, and you do not need to write a whole chapter. Fixing a typo, sharing what worked on a real project, or translating a page all count. This page walks you through every way to contribute, from a one-line fix to a full chapter, step by step.

Ways to contribute

- **Fix or improve a page:** correct an error, clarify a sentence, add a missing reference.
- **Write a chapter or section:** fill a gap the playbook does not yet cover.
- **Share a case study:** what you did on a real project, including what went wrong.
- **Add an example:** a real, verifiable dataset or paper that illustrates a point.
- **Translate a page:** into Hausa, Amharic, Swahili, French, Portuguese, or another language.
- **Open a discussion:** ask a question or challenge an approach. Disagreement makes the guide better.

Everything lives in one repository: github.com/warakacommunity/playbook.

Pick the path that fits your change

There are three ways to make a change, from the quickest to the most hands-on. Choose by how big your change is, not by how experienced you are.

If you want to...	Use...	Needs
Fix or edit text on a page	The online editor (below)	A GitHub account
Make a small change to one file	GitHub's web editor	A GitHub account
Add a chapter or change several files	Fork and pull request	Git on your computer

Path 1: Edit on the site (easiest)

Best for fixing typos, rewording a sentence, or adding a link. You never leave the browser.

1. Open the **online editor**.
2. Sign in with GitHub when prompted. One click authorises you securely on GitHub, so nothing is ever typed into this site. This lets us credit your work and open the change under your name.
3. Find the page you want to change and make your edit.
4. Add a short note describing what you changed, then submit.

The site opens a pull request for you automatically. A maintainer reviews it and merges. That is the whole process: no git, no setup.

i NOTE

If **Contribute** opens a new tab on afriplaybook.waraka.org, that is expected. GitHub sign-in runs on the project's editing site. It opens the same chapter you were reading, so just sign in and edit there.

Path 2: Edit one file on GitHub

Best for a small change when you would rather work on GitHub directly.

1. Find the file in the repository. Chapters live under [docs/](#).
2. Click the **pencil icon** (Edit this file) at the top right of the file.
3. Make your change. GitHub creates a fork for you if you do not have write access.
4. At the bottom, write a short description and click **Propose changes**.
5. Click **Create pull request**.

Path 3: Fork and pull request (for chapters and larger changes)

Best for adding a new chapter or editing several files at once. This needs [git](#) and [Node.js 18+](#) on your computer. Even if you are new to git, you can follow these steps.

Step 1: Open an issue first

Before writing a chapter, [open an issue](#) describing what you plan to add. This avoids two people writing the same thing and lets maintainers point you in the right direction.

Step 2: Fork and clone

Open [the repository](#) and click **Fork** (top right). Then clone your fork:

```
git clone https://github.com/<your-username>/playbook.git
cd playbook
```

Step 3: Install and run locally

You need Node.js 22 and Yarn 1.x (install it with `npm install -g yarn`). Use Yarn, not npm, so you get the exact versions the project is tested with.

```
yarn install --frozen-lockfile
yarn start      # opens a live preview at http://localhost:3000
```

The preview reloads as you edit, so you can see your change immediately.

Step 4: Create a branch

Never work on `main` directly. Create a branch named for your change:

```
git checkout -b chapter/your-topic-slug
```

Step 5: Add or edit your content

Chapters are Markdown files under `docs/`, grouped into folders by topic. To add a page, create a new `.md` file in the right folder and start it with frontmatter:

```
---
sidebar_position: 3
---

# Your Chapter Title

Your content here.
```

`sidebar_position` controls where the page appears in the sidebar. Pick the number for the slot you want, and bump the pages after it if needed.

Step 6: The sidebar updates itself

The sidebar is generated automatically from the folder structure and each page's `sidebar_position`, so there is nothing extra to edit. To rename a folder's label, edit its `_category_.json`.

Step 7: Preview your change

Check your page in the running preview (`http://localhost:3000`). Read it on a narrow window too; most contributors read on a phone.

Step 8: Run the build

This catches broken links and other errors before you open a pull request:

```
yarn build
```

Fix anything it flags. A clean build is the main thing reviewers check for.

Step 9: Commit and push

```
git add .
git commit -m "Add chapter on <your topic>"
git push origin chapter/your-topic-slug
```

Step 10: Open a pull request

1. Go to your fork on GitHub and click **Compare & pull request**.
2. Write a short description of what you added and why. Link the issue from Step 1.
3. Click **Create pull request**.

A maintainer will review it, suggest any changes, and merge it once it is ready.

Writing guidelines

A few things every contribution should follow:

- **Write plainly.** Short sentences, active voice, one idea per paragraph. Explain jargon the first time you use it.
- **Cite real sources.** Every claim and example must point to a real, verifiable paper, dataset, or project, never an invented or unchecked citation.

For how to structure a page, add sections and subsections, and use the colored callout boxes (in **Markdown or Word**), see [How to Write the Document](#). For the full style rules and deeper repository details, see [CONTRIBUTING.md](#).

Translating a page

Translations are community-maintained. Use the language switcher in the top-right of the navbar to see which languages exist, and see [CONTRIBUTING.md](#) for how the translation files are organised.

Get help

- **Questions or ideas:** [GitHub Discussions](#)
- **Chat with the community:** [Discord](#)
- **Found a bug:** [open an issue](#)

See everyone who has contributed so far on the [Contributors](#) page.

 [Cite this page](#) ⌚ 5 min read

CONTRIBUTORS  (<https://github.com/shmuhammad>)  (<https://github.com/abumafri>)  (<https://github.com/seyyaw>)

Our contributors

The AfriPlaybook is written and maintained by volunteers: researchers, students, linguists, translators, annotators, and community organisers. Thank you to everyone who has helped build it. Below are the people who have contributed to the [playbook repository](#). The list updates each time the site is published.



@seyyaw

MAINTAINER

(<https://github.com/seyyaw>)



@shmuhammadd

MAINTAINER

(<https://github.com/shmuhammadd>)



@abumafrim

MAINTAINER

(<https://github.com/abumafrim>)



@Tadesse-Destaw

MAINTAINER

(<https://github.com/Tadesse-Destaw>)



@keduog

(<https://github.com/keduog>)



Ibrahim Ahmad

Not listed here?

Much of the work behind a guide like this never appears on GitHub: reviewing chapters, translating pages, annotating data, and organising people. If you have helped in one of these ways, open an [issue on GitHub](#) or tell us on [Discord](#) so we can credit you.

If you have committed to the repository but are missing from the list, check that the email address you commit with is linked to your GitHub account.

To join the list of contributors, see [How to contribute](#).

 [Cite this page](#)  1 min read

CONTRIBUTOR  (<https://github.com/shmuhammadd>)

How to Write a Chapter

The [How to Contribute](#) section covers the requirements to make contributions, namely editing on the site, forking, opening a pull request. In this Section, we cover the writing itself, how to structure a page, add sections and subsections, and use the colored callout boxes (admonitions) that make a page easy to read.

You can write in **two ways**, but there are more options (see the [How to Contribute](#) section):

- **Markdown:** This is the native format of the playbook. What you type is what gets published, but using the markdown syntax. If you are new to markdown, it is not complicated, we provided basics below, but you can also get started [here](#). This is the best option if you are comfortable editing text files or using the online editor.
- **Microsoft Word:** This is a familiar editor for you to draft your chapter, then we have an internal maintainer that converts it to Markdown. Best for long chapters or contributors who do not work with Markdown.

Both produce the same published page. The rules below are written so a Word draft and a Markdown draft turn into identical output.



THE GOLDEN RULE

Do not type the chapter title or chapter number yourself. The playbook adds the title and number (`# 1.` [Introduction](#)) automatically from the chapter's place in the sidebar. Your first heading is the page's *first section*, not its title. In Markdown that is `##` ; in Word that is **Heading 1**.

Writing in Markdown

Sections and subsections

Headings build the structure of the page and the table of contents on the right. Because the chapter title is added for you, you start one level down:

```
## A section           ← top-level section (shows in the right-hand menu)
### A subsection       ← nested under the section above
#### A sub-subsection ← use sparingly; deep nesting is hard to read
```

Rules that keep pages clean:

- **Never use a single `#`.** That is the chapter title, and the playbook owns it.
- **Don't skip levels.** A `###` is followed by `####`, not `#####`.
- **Keep headings short.** They become menu entries; a full sentence breaks the layout.
- **One idea per section.** If a section runs past a screen or two, split it.

Admonitions: the colored callout boxes

Admonitions are the boxed callouts (like the green "golden rule" box above) that pull a reader's eye to something important. Wrap text in `:::` fences:

```
:::note
Neutral aside or extra context. Gray.
:::

:::tip
Best practice, or what worked on a real project. Green.
:::

:::info
Something the reader needs to know first: a prerequisite or definition. Blue.
:::

:::warning
A common mistake or gotcha. Yellow.
:::

:::danger
Something irreversible or unsafe: data loss, consent, security. Red.
:::
```

Give a box your own heading by putting it in square brackets:

```
:::tip[Contribute]
You don't need to write a whole chapter to help.
:::
```

Rules:

- Leave a **blank line before and after** each `:::` fence and around the content inside.
- You can put normal Markdown inside: **bold**, links, lists, code.
- Use them sparingly: **at most one per section**, or the page reads like a warning label.

When to use which box

Box	Use it for
<code>tip</code>	Recommended practice, "what worked on a real project"
<code>note</code>	Side context, optional detail
<code>info</code>	Prerequisites, definitions, version notes
<code>warning</code>	Common mistakes, gotchas
<code>danger</code>	Data loss, irreversible steps, consent, security

Other things that make a page nicer

- **Code blocks:** always set the language (such as **python**, **java**) so it gets highlighted, and add a title if helpful:

```
python title="train.py"
print("hello")
```

This is important specially for chapters that talks about technical aspects such as data cleaning, visualization and so on.

- **Images:** put them in the chapter's `images/` folder and always write alt text describing the figure:

```
![AfricaNLP papers grew roughly fourteenfold between 2006 and 2024.](images/africanlp-growth.svg)
```

- **Links:** link to other pages with a relative path so the build can check them: `[Core principles](./core-principles)`.
- **Collapsible detail:** hide long, optional content:

```
<details>
  <summary>Full configuration example</summary>

  ...content...

</details>
```

- **Tables:** keep them narrow. Most contributors read on a phone, and wide tables break the layout.
- **No em-dashes:** don't use the long dash. Use a colon, a comma, parentheses or a new sentence instead.

Writing in Microsoft Word

You can draft in **Word** and let a maintainer convert it. To make the conversion clean, use Word's **built-in styles** (the Heading 1 / Heading 2 buttons), not manual bold text or bigger fonts. The converter only understands real styles.

Headings map one level down

Because the playbook adds the chapter title automatically, **Word's Heading 1 becomes a section, not the title**. Use this mapping:

In Word, use...	Becomes in the playbook	Markdown equivalent
<i>(don't add one)</i>	The chapter title + number	<code># 1. Title</code>
Heading 1	A top-level section	<code>##</code>
Heading 2	A subsection	<code>###</code>
Heading 3	A sub-subsection	<code>####</code>
Normal / Body text	A paragraph	plain text
Bulleted / Numbered list	A list	<code>-</code> / <code>1.</code>

So: don't type a chapter title at the top of your Word file. Start straight into your first **Heading 1** section.

Admonitions in Word

Word has no built-in admonition, so use the convention below. The simplest, most reliable way is to **type the fences as plain text**, each on its own line, in a normal (Body) paragraph, *not* styled as a heading:

```
::: tip[Contribute]
You don't need to write a whole chapter to help.
:::
```

When the document is converted, those lines survive as-is and the playbook renders them as a real colored box. Keep a blank paragraph before and after the fences, exactly as in Markdown.

A few more Word tips

- **Images:** paste them inline where they belong, and write a one-line caption underneath each so it can become alt text.
- **Links:** use Word's real hyperlinks (Insert → Link), not bare pasted URLs, so they convert correctly.
- **Don't fake structure:** avoid manual numbering, hand-drawn boxes, or extra blank lines for spacing. Let styles do the work; the layout is applied for you on the site.

Preview before you submit

Whichever format you write in, check how it looks before opening a pull request. The most accurate preview is the live site itself:

```
npm start # opens a live preview at http://localhost:3000
```

The preview reloads as you edit, so admonitions, headings, and images appear exactly as they will once published. Read the page on a narrow window too, since most readers are on a phone.

For the full style rules, see [CONTRIBUTING.md](#).

 [Cite this page](#)  5 min read

CONTRIBUTORS  (<https://github.com/abumafri>)  (<https://github.com/shmuhammad>)  (<https://github.com/seyyaw>)

Core principles

A handful of beliefs run through every chapter of this playbook. They are the reason it gives the advice it does. If you remember nothing else, remember these.

Build with speakers, not just for them

The people who speak a language know what is correct, what sounds natural, and what is offensive. They belong at the centre of dataset work, not at the end of a pipeline as cheap labellers. The [participatory model](#) (Nekoto et al., 2020) that this playbook grows out of treats native speakers as co-creators, and it is the approach we recommend throughout.

Consent and ownership come first

Data comes from people, and those people keep a stake in it. Ask for informed consent before anyone contributes, be honest about how the data will be used, and make sure the community stays the owner of what it helps create. Ownership is not a box to tick at release. It shapes every decision, from what you collect to who you let use it.

Quality over scale

A small, clean, well-documented dataset is worth more than a large, noisy one. Web-scale crawls for low-resource languages are full of mislabelled and machine-translated text ([Kreutzer et al., 2022](#)). It is better to label a thousand examples well than a million badly. The whole [data quality](#) chapter exists because this principle is easy to say and hard to keep.

Context travels with the data

A dataset without documentation is hard to trust and easy to misuse. Record what you collected, from whom, how, and why. Name the languages, scripts, and varieties precisely. Good [documentation](#) is what lets other people reuse your work, and what lets you defend it later.

Build in the open, design for reuse

Release under a clear, open licence, and make the data, the guidelines, and the process reproducible. Work that others can find, check, and extend compounds over time. Work locked away helps no one and is usually lost.

Do no harm

Protect the people in the data and the people who build it. Watch for content that is offensive, that identifies someone, or that is unsafe to release. When you are unsure, ask the community, and err on the side of caution.

These principles do not always pull in the same direction. Scale competes with quality; openness can sit uneasily with privacy. Much of this playbook is about making those trade-offs well, in the open, and with the people whose languages are at stake.

 [Cite this page](#)  2 min read

CONTRIBUTOR  (<https://github.com/abumafrim>)

What this playbook is (and isn't)

The one-sentence framing

AfriPlaybook is the canonical decision framework for anyone starting an African-language NLP project, grounded in the empirical experience of Masakhane's own past projects.

Everything the playbook adds should sharpen that sentence. Everything it removes should be justified against it.

What this playbook is not

- It is **not** a tutorial on training or fine-tuning models. That work is well covered by the [Hugging Face NLP Course](#), the [Google Deep Learning Tuning Playbook](#), the [Google Text Classification guide](#), and [Advanced NLP with spaCy](#). We link to them; we do not rewrite them.
- It is **not** a textbook. For the theory (tokenisation, sequence models, evaluation metrics, statistical foundations), go to Jurafsky and Martin's [Speech and Language Processing](#). It is authoritative and free.
- It is **not** a catalogue of code recipes. Code drifts out of date faster than any playbook can be maintained. When we point at code, we point at a live, versioned, maintained repository. When we ship code ourselves, it lives in a companion repo that ships its own release tags.
- It is **not** neutral. We take positions. Character-level metrics for morphology-rich languages ([chrF](#), CER) over word-level ones (BLEU, WER). Per-language and per-class score reporting over a single average. Human evaluation for anything generative. Consent from the people in the data. Community ownership of the resources built. If you disagree with those positions, this is not the playbook for you.

What this playbook is

Structurally, this playbook belongs to the genre of **dataset-lifecycle and responsible-AI handbooks**. Its closest cousins are [Datasheets for Datasets](#) (Gebru et al., 2018), [Data Statements for NLP](#) (Bender & Friedman, 2018), [Model Cards](#) (Mitchell et al., 2019), and [The Turing Way](#). What we add on top of that lineage:

- A **first-class annotation-workforce chapter** (how to design work, hire, brief, review, pay, and retain annotators), which most dataset-lifecycle handbooks touch only lightly.
- **First-class modality tracks** for African-language reality: text, speech (ASR, TTS, S2ST), OCR/document AI, sign language and video. Most NLP playbooks are text-only.
- An **explicit low-resource, African-context lens**: patchy connectivity, multiple scripts, code-switching, small compute budgets, community-led workflows, community IP concerns.

Substantively, our job is to be the one document a new project can read that saves it a quarter of duplicated work.

Why this scope: the evidence

In July 2026 we commissioned an independent research pass comparing AfriPlaybook against twenty candidate "playbooks" across three groups: canonical model-and-training NLP playbooks (Google DL Tuning, Google Text Classification, Hugging Face NLP Course, spaCy), data / responsible-AI handbooks (Datasheets, Data Statements, Model Cards, Turing Way, PAIR, BigScience ROOTS, Deon), and low-resource work AfriPlaybook already sits alongside (MasakhaneMT, MasakhaNER 1 and 2, AfriSenti, AfriQA, LAFAND-MT, FLORES, NLLB). The verdict was clear:

Eight of ten numbered chapters center on data collection, annotation workforce, quality assurance, governance, documentation, and community. Only the model-building chapter touches modelling, and it is a short chapter. The canonical Group-A NLP playbooks are structured around model choice, fine-tuning, hyperparameter tuning, and deployment, and explicitly exclude annotation, low-resource languages, and cultural considerations. AfriPlaybook is structurally a data/annotation-handbook, not a model-tuning-playbook.

That is not a criticism. It is a positioning. The playbook is what those Group-A playbooks assume has already been done. Trying to compete with them on their own territory would spread AfriPlaybook thin and produce content that dates within a year. Owning the annotation, data quality, evaluation, community, and African-context deployment space is defensible, durable, and, for the community the playbook serves, genuinely load-bearing.

What comes next: the phased plan

The current playbook is a solid draft of the dataset-lifecycle spine. The next expansion is about making the playbook **the shortest path to a working project** for the reader who has a task and a language and doesn't know where to start. In priority order:

Phase 1: the two highest-leverage additions

- **Before You Start.** A four-step method for checking prior work before building a dataset: search for what exists, judge it against a checklist, decide whether to reuse, extend, or build, and learn from the people who built it. Three worked examples (NER, sentiment, and hate speech) apply the method to well-known datasets. They are examples, not complete lists, because no hand-written list can keep up with the field.
- **Case Studies:** real retrospectives from Masakhane's own body of work: MasakhaNER 1 and 2, AfriSenti, AfroBench, LAFAND-MT, AfriQA. Each answers the same set of questions (team size, timeline, budget, hardest problem, biggest mistake, what surprised the team) from the people who did the work. Two case studies of this quality are worth more than ten chapters of general prose.

Phase 2: cross-cutting realism

- **Deployment for African contexts.** Offline and patchy-connectivity NLP. Edge devices. SMS/USSD/WhatsApp as NLP surfaces. Multilingual switching in a single conversation. Non-standard scripts (Ajami, Ge'ez, N'Ko) in real UIs.
- **Cross-language transfer matrix.** Empirical guidance, drawn from Masakhane's own papers, on which African languages benefit from which transfer sources. Bantu-to-Bantu, Cushitic-to-Cushitic, script transfer where relevant.

Phase 3: durable governance

- **Legal, consent, and community IP.** Country-level data laws (Nigeria NDPA, Kenya DPA, South Africa PoPIA, Ghana DPA, Rwanda) and their practical implications. Consent frameworks that work with non-literate speakers. Community IP frameworks. Anti-extraction release patterns.
- **Long-tail language onboarding.** For a language with a speaker community but no digital corpus, what is step one? Orthography, corpus bootstrapping, community mobilisation, anti-patterns.

Phase 4: compute-poor practice

- **Compute-poor training and evaluation.** Community-GPU, Colab, Kaggle patterns. LoRA and QLoRA (linked to canonical, not re-derived). Distillation for edge deployment. When cross-language transfer is a compute-saving strategy.

Phase 5: finding current resources (redesigned)

The original Phase 5 was a curated living directory: every African-language dataset, model, tool, and benchmark, with a one-line editorial opinion on each entry, updated on a semiannual review cadence. On honest reflection, the review cadence was the load-bearing part, and no team has capacity to eyeball 200+ entries semiannually and give each a fresh editorial line. A stale directory that says "SOTA as of 2026-Q3" in 2027 is worse than no directory.

The redesigned Phase 5 ships as [Step 1 of Before You Start](#), which names a few places to start searching (the ACL Anthology, Lanfrica, Hugging Face, the AfricaNLP workshop proceedings, Google Scholar, and arXiv) and teaches search patterns that find current African-language datasets. It does not list datasets; the sources list themselves.

Tradeoff acknowledged: we give up per-entry editorial opinion on every dataset and model (readers learn to judge entries themselves with the checklist in [Before You Start](#)). We gain a chapter that stays honest for years without touch and does not require a named maintainer to unblock.

What we deprioritise

- **Hyperparameter tuning methodology.** Owned by Google's DL Tuning Playbook. We link.
- **Model architecture selection recipes.** Owned by the Hugging Face NLP Course and the model cards on the HF Hub. We link.
- **Fine-tuning recipes for named base models.** Same. We link.
- **Prompt engineering catalogues.** Fast-moving. Owned by the model providers' own cookbooks. We link.
- **Serving, latency, and cost optimisation for cloud deployments.** Owned by MLOps playbooks and cloud providers. We may cover the *African-context* pieces (edge, offline) but do not compete on the general case.

The one rule that keeps this playbook honest

Every chapter is graded against one question: does a reader with a specific task and language leave with an actionable next step in fifteen minutes or less?

If yes, the chapter earns its place. If no, it is either rewritten to answer that question, or removed. Comprehensive without useful is the failure mode we most fear.

This decision record will be revisited at each major playbook release. See the [changelog](#) for the record of what changed and when.

 [Cite this page](#)  6 min read

CONTRIBUTORS



(<https://github.com/shmuhammadd>)



(<https://github.com/abumafrim>)



(<https://github.com/seyyaw>)

Before You Start

Teams sometimes spend months collecting data that already exists. A team may collect Hausa sentiment data that [AfriSenti](#) already covers, or annotate Yoruba named entities before discovering [MasakhaNER 2](#).

This happens because African-language datasets are hard to find, and the same language often appears under different names and codes. For example, Swahili has at least three codes in common use:

- `sw`: the two-letter ISO 639-1 code.
- `swa`: the ISO 639-3 code for Swahili as a macrolanguage, covering all its varieties.
- `swh`: the ISO 639-3 code for Coastal (Standard) Swahili alone.

All three mean Swahili, but a computer treats them as three different labels, and projects do not agree on which to use. The Masakhane benchmarks (AfriMMLU, AfriXNLI, AfriMGSM) use `swa`. FLORES and Belebele use `swh`. Many Hugging Face datasets and web-crawled corpora use `sw`. If you search for one code, you miss the data filed under the others.

A complete list of African-language datasets would not solve this, because new datasets appear every month and no list stays current for long. Instead, this chapter describes four steps to take before you build: search for what exists, check whether it fits your needs, decide whether to reuse, extend, or build, and learn from the people who made it.

Step 1: Search for prior work

Search for your language and task together (for example, "Hausa sentiment") to find datasets you might reuse as they are. Plan a few hours for these searches, and keep notes as you go.

Where to look first:

- [ACL Anthology](#): the main archive of NLP research papers, from the major conferences, journals, and workshops. Search for your language and task, then check each paper for a released dataset. For a larger, systematic search, the Anthology's [Python package](#) (`pip install acl-anthology`) gives you the metadata of every paper to filter yourself.
- [Lanfrica](#): a continually updated catalogue of African language resources, including datasets, models, and papers.
- [AtlasNLP](#): a map of more than 13,000 NLP datasets described in ACL Anthology papers, which you can filter by task, language, and country. Use it to see which datasets already exist for your language or country.
- [Hugging Face datasets, filtered by language](#): replace `swa` with your language's code (`hau` Hausa, `yor` Yoruba, `amh` Amharic, `zul` isiZulu). Add a task filter, such as `text-classification`, to narrow the results.
- [AfricaNLP workshop proceedings](#): papers from the annual workshop on African-language NLP. Skim the latest year to see what is new.
- [Google Scholar and arXiv](#): new work often appears here first. Combine your language, task, and a date range, for example `"Hausa" sentiment dataset 2023..2026`.

How to search well:

- **Try every name and code the language goes by:** its English name, its own name for itself, alternative spellings, and all of its ISO codes (for Swahili: `sw`, `swa`, and `swh`). Older papers may use names the community no longer uses.

- **Search for related languages too.** A dataset for a closely related language can still give you guidelines, a label set, or a starting point.
- **Ask.** Post a short question in the [Masakhane](#) community or a language-specific group: *"Does anyone know of X data for Y?"* Much work is unpublished or hard to find.

Keep a simple record of everything you find: name, link, task, languages, size, licence, and whether guidelines were released. You will need it for Step 2, and it becomes the related-work section of your own paper.



TEMPLATE: DATASET SEARCH LOG

A fill-in log that follows the four steps on this page: your language's names and codes, every search you ran, a Pass, Partly, or Fail mark for each dataset against the Step 2 checklist, your Step 3 decision, and the people you contacted. [Open the dataset search log template](#) or [download it as a Word document](#).

Step 2: Judge what you find

Check each dataset you find against this list:

- **Licence.** Does it allow your use? Many African-language datasets are released for non-commercial use only (CC BY-NC).
- **Access.** Can you download it today, or is it "available on request" from an author who no longer replies? Are you looking at the latest version?
- **Fit.** Does it match your language variety, dialect, script, and domain? News text does not stand in for social media or conversational speech.
- **Labels.** Is the label set right for your task, or would you need to relabel?
- **Splits.** Is the data divided into fixed training, development, and test sets, so that results can be compared across papers? If the test set is public, large language models may have been trained on it, which makes their scores look better than they are.
- **Guidelines.** Were the annotation guidelines released? You can reuse good guidelines even if you cannot use the data.
- **Quality.** Who annotated it: native speakers or crowd workers? Is inter-annotator agreement reported? It measures how often annotators chose the same label, and low agreement is a warning sign.
- **Documentation.** Is there a datasheet or data card: a document that describes how the data was collected and what it contains? Is consent described?

A dataset that fails several of these checks may still be worth studying, even if you cannot build on it.

Step 3: Reuse, extend, or build

Use your notes from Steps 1 and 2 to choose what to do. Find the row that matches what your search turned up and how it did against the Step 2 checklist, then follow that row across.

What you found	Your choice	What to do
A dataset for your language and task that passes the Step 2 checks	Reuse it	Do not build a new one. Spend your effort on what is still missing, such as evaluation, a new domain, or a deployment.
A dataset for your language and task that partly fits (wrong domain, missing labels, or a different dialect)	Extend it	Add the missing domain, labels, or dialect. Reuse the original guidelines, keep your data splits compatible with theirs, and tell the original authors.
A dataset for your language and task that fails the checks (unusable licence, poor quality, or unavailable)	Build a new one, learning from the old one	Read its paper and guidelines before you start, and explain in your documentation why you could not reuse it.
A dataset only for a related language , or for your language on a related task	Borrow, then build small	Borrow its guidelines and label set. Try cross-language transfer first, then build a small, high-quality evaluation set in your language.
Nothing relevant	Build from scratch	Start with the long-tail language onboarding chapter, then Data Collection , Annotation Design , and Data Quality .

If more than one row applies, choose the one nearest the top. Reusing or extending data usually costs less than building it.

Step 4: Learn from the people who built it

Papers describe what worked. They rarely describe what went wrong, what took longer than planned, or what the team would change.

Read first, in this order: the paper's limitations section, the annotation guidelines, the datasheet, and the open issues on its repository.

Then contact the authors. Many are happy to help. Ask specific questions:

- What would you do differently if you started again?
- What took longer or cost more than you expected?
- How did you recruit, train, and pay annotators?
- Is there anything you collected but did not release?
- Would you be interested in collaborating on an extension?

Then pass it on. When your own project is done, write up what you learned with the [retrospective template](#) and add it to the [Case Studies](#).

Worked examples

The three pages below apply these four steps to real tasks, using well-known datasets as examples. If your language or task is missing, search for it using Step 1.

- [Named Entity Recognition](#): a well-covered task, where the usual answer is to reuse or extend.
- [Sentiment analysis](#): where domain and culture decide whether existing data fits.

- [Hate speech and content safety](#): where you must also protect annotators and users.

Each page shows the date it was last reviewed. Newer datasets may have appeared since then.

 [Cite this page](#) ⌚ 7 min read

CONTRIBUTORS



(<https://github.com/abumafrim>)



(<https://github.com/shmuhammadd>)



(<https://github.com/seyyaw>)

Named Entity Recognition

Worked example. Last reviewed: 2026-07-07.

This page applies the [four steps from Before You Start](#) to named entity recognition (NER): finding people, organisations, places, and dates in text. It uses **MasakhaNER** as the example because it is the best-documented NER effort for African languages. If your language is not mentioned here, go back to [Step 1](#) and search.

Step 1: What a search turns up

Search Hugging Face, Lanfrica, or the ACL Anthology for NER in almost any major African language and you will quickly reach two datasets:

- [MasakhaNER 2](#) (Adelani et al., 2022): 20 languages. Bambara, Chichewa, chiShona, Ewe, Fon, Ghomala, Hausa, Igbo, isiXhosa, isiZulu, Kinyarwanda, Kiswahili, Luganda, Luo, Mossi, Nigerian Pidgin, Setswana, Twi, Wolof, and Yoruba.
- [MasakhaNER 1](#) (Adelani et al., 2021): the first 10 languages. Version 2 covers most of them, but version 1 is still the one to use for **Amharic**, which version 2 does not include. Its paper also explains how the annotation process was designed.

The data is on the [Hugging Face Hub](#), and the [GitHub repository](#) has the annotation guidelines and evaluation scripts.

You will also find **automatically labelled ("silver") NER data** that covers many more languages, built by projecting labels from Wikipedia links rather than by human annotators. It is tempting because of its coverage. Step 2 is where you find out whether its quality is good enough for your purpose.

Step 2: Judging MasakhaNER 2 against the checklist

- **Licence.** CC BY-NC 4.0: free for research, **not for commercial use**. If you are building a product, check this first.
- **Access.** Openly downloadable from Hugging Face and GitHub.
- **Fit.** News text. Social media, government documents, or speech transcripts look different, so expect lower accuracy on them.
- **Script.** Latin script only. Hausa Ajami, Arabic-script Wolof, and Ge'ez-script languages are not covered.
- **Code-switching.** The sentences are monolingual. Real text often mixes languages within a sentence (Hausa–English, Swahili–English, French–Wolof). If yours does, expect results to be worse than the published numbers.
- **Labels.** Person, organisation, location, and date. "Organisation" may not match what your use case means by it, so read the guidelines before assuming the label set fits.
- **Splits.** Fixed train, development, and test splits, so results are comparable across papers.
- **Guidelines.** Released. Even if your language is not covered, the guidelines are the best starting point for writing your own.
- **Quality.** Annotated by native speakers through a participatory process. The papers describe it in detail.
- **Diacritics.** In Yoruba, Igbo, and several other languages, tone marks carry meaning. Check that your text processing keeps them, or entity boundaries will shift and results will look worse than they are.

Step 3: Reuse, extend, or build

Find the row that matches your situation, then follow it across. The first question is whether your language is covered by MasakhaNER 2.

What you found	Your choice	What to do
Your language is covered by MasakhaNER 2, and its labels (person, organisation, location, date) are enough for your task	Reuse MasakhaNER 2	Do not build a new corpus for these labels.
Your language is covered , but you need extra entity types (money, products, culture-specific)	Extend it	Annotate the extra types on the existing splits and reuse the guidelines. Do not re-annotate person, organisation, and location; merge.
Your language is not covered , but it is related to one of the 20 (same family or script)	Borrow, then build small	Try cross-language transfer from the closest covered language, then build a small evaluation set in your own language using the MasakhaNER guidelines.
Your language is not covered and not related to any of the 20	Build from scratch	Start with the long-tail language onboarding chapter, then Data Collection , Annotation Design , and Data Quality . Read the MasakhaNER guidelines before writing your own, and aim for 5,000–10,000 sentences to start.

Step 4: Learning from the people who built it

MasakhaNER was built by volunteer researchers and native-speaker annotators across Africa. Their way of organising the work is worth studying before you design your own project.

Read first:

- The [MasakhaNER case study](#) in this playbook: how the project was organised and what the team learned.
- The [MasakhaNER 2 paper](#) (Adelani et al., 2022): the 20-language dataset and its analysis.
- The [MasakhaNER 1 paper](#) (Adelani et al., 2021): how the annotation process was designed.
- The [participatory research paper](#) (Nekoto et al., 2020): the Masakhane way of working that MasakhaNER grew out of.

Questions to ask the authors:

1. How did you recruit and coordinate annotators in 20 countries?
2. Where did annotators disagree most, and how did you resolve it?
3. What changed between version 1 and version 2, and why?

Once you have your data, training a model is well covered elsewhere. The [Hugging Face token classification tutorial](#) is a good start.

 [Cite this page](#)  4 min read

CONTRIBUTORS



(<https://github.com/abumafrim>)



(<https://github.com/shmuhammadd>)

Sentiment analysis

Worked example. Last reviewed: 2026-07-07.

This page applies the [four steps from Before You Start](#) to sentiment analysis: deciding whether a text is positive, negative, or neutral. It uses **AfriSenti** and **NaijaSenti** as the examples because they are the largest community-built sentiment datasets for African languages, annotated by native speakers. If your language is not mentioned here, go back to [Step 1](#) and search.

Step 1: What a search turns up

Search Hugging Face, Lanfrica, or the ACL Anthology for sentiment in almost any major African language and you will quickly reach two datasets:

- **AfriSenti** [\(Muhammad et al., 2023\)](#): more than 110,000 tweets in 14 languages. Amharic, Algerian Arabic, Hausa, Igbo, Kinyarwanda, Moroccan Arabic, Mozambican Portuguese, Nigerian Pidgin, Oromo, Swahili, Tigrinya, Twi, Xitsonga, and Yoruba. It was also used in the [AfriSenti-SemEval 2023 shared task](#), so many published systems report results on it.
- **NaijaSenti** [\(Muhammad et al., 2022\)](#): the earlier Nigerian dataset that AfriSenti grew from. Hausa, Igbo, Nigerian Pidgin, and Yoruba, with around 30,000 tweets per language (about 14,000 for Nigerian Pidgin). Its paper also released sentiment lexicons.

AfriSenti is on the [Hugging Face Hub](#), and the [GitHub repository](#) has the data, the evaluation script, and individual annotator labels for several languages. NaijaSenti is on the Hub as [HausaNLP/NaijaSenti-Twitter](#).

You will also find **copies of the same data** uploaded by other accounts, and small **sentiment datasets built by translating English data** or labelling tweets automatically with emojis or keywords. Copies are tempting because they load easily. Translated and auto-labelled data is tempting because it is cheap. Step 2 is where you check the licence of the copy you actually download, and whether auto-generated labels reflect how people in your language express feeling.

Step 2: Judging AfriSenti against the checklist

- **Licence.** Check the copy you use. The main Hugging Face card says CC BY 4.0, but other copies on the Hub are tagged CC BY-NC-SA (non-commercial, share-alike). If you are building a product, confirm the licence with the authors first.
- **Access.** Openly downloadable from Hugging Face and GitHub.
- **Fit.** Twitter/X text only. It is short, informal, and full of abbreviations and hashtags. Customer service transcripts, health surveys, formal writing, or long comments look very different. Always build an evaluation set from your own target domain.
- **Culture.** What reads as negative in one culture can read as neutral or even polite in another. Indirect criticism is common in many African-language contexts and is often labelled neutral by annotators from outside the culture. Read the paper's annotation section before writing your own guidelines.
- **Code-switching.** Normal, not an edge case. Tweets in every covered language mix in English, French, Arabic, or a regional lingua franca. Your data will too.

- **Script.** Amharic and Tigrinya are in Ge'ez script. Algerian and Moroccan Arabic tweets may appear in Arabic or Latin script. Check that your text processing handles both.
- **Labels.** Three classes: positive, negative, neutral. Neutral is the hardest class. Sarcasm, mixed feelings, indirect expression, and plain facts all end up there. Report results per class, not only an overall score (see [core principles](#)).
- **Label scope.** Polarity is not emotion, stance, polarization, or harm. If you need emotion labels, see [emotion analysis](#). If you need content moderation, see [hate speech analysis](#). If you need to detect hostility between groups, look at polarization data such as [POLAR](#). Do not reuse AfriSenti labels for these tasks.
- **Splits.** Fixed train, development, and test splits for every language, so results are comparable across papers. The test sets have been public since 2023.
- **Guidelines.** Partial. The repository has a guidelines file for Algerian Arabic only. The paper describes the annotation process for each language.
- **Quality.** Annotated by native speakers. Individual annotator labels are released for several languages, so you can measure agreement yourself.
- **Privacy.** The tweets were anonymised: user mentions were replaced with `@user` and URLs were removed. If you collect new social media data, do the same, and check the platform's terms before you redistribute it.
- **Transfer.** Sentiment depends on culture-specific words and idioms. It transfers less well between languages than structural tasks such as NER. Measure transfer before you rely on it.

Step 3: Reuse, extend, or build

Find the row that matches your situation, then follow it across. The first question is whether your language is covered by AfriSenti or NaijaSenti.

What you found	Your choice	What to do
Your language is covered by AfriSenti (14 languages) or NaijaSenti (4 Nigerian languages), and Twitter/X text is close to your domain	Reuse AfriSenti or NaijaSenti	Do not build a new corpus. Check the licence, then spend your effort on a careful human review of results on your own data.
Your language is covered , but your domain is different (formal writing, market speech, health, customer service)	Extend it	First build a 500–2,000 example evaluation set from your own domain. Only if AfriSenti does not work well enough, add a small 2,000–5,000 example in-domain training set, using the same three labels.
Your language is not covered , but it is related to a covered language	Borrow, then build small	Try cross-language transfer from the closest covered language, then build a small evaluation set in your own language. Sentiment transfers weakly, so measure before you rely on it.
Your language is not covered and not related to a covered language	Build from scratch	Start with the long-tail language onboarding chapter, then Data Collection , Annotation Design , and Data Quality . Read the AfriSenti paper before writing your guidelines, and start with a pilot of 2,000–5,000 examples.

Step 4: Learning from the people who built it

AfriSenti was built by native-speaker researchers and annotators in many African countries, working in a participatory way.

Read first:

- The [AfriSenti paper](#) (Muhammad et al., 2023): the 14-language dataset, especially the section on the challenges of curating each language.
- The [NaijaSenti paper](#) (Muhammad et al., 2022): how tweets in four Nigerian languages were collected, filtered, and labelled.
- The [participatory research paper](#) (Nekoto et al., 2020): the Masakhane way of working that AfriSenti follows.

Questions to ask the authors:

1. How did you decide what counts as neutral, and where did annotators disagree most?
2. How did you find enough tweets in languages with little Twitter activity?
3. What would you change about the guidelines if you started again?

Further reading

- [Quality at a glance \(Kreutzer et al., 2022\)](#): an audit of web-crawled low-resource data. Read it before you trust scraped or automatically labelled sentiment data.

Once you have your data, training a model is well covered elsewhere. The [Hugging Face sequence classification tutorial](#) is a good start.

 [Cite this page](#)  5 min read

CONTRIBUTORS<https://github.com/abumafrim><https://github.com/shmuhammad>

Hate speech and content safety

Worked example. Last reviewed: 2026-07-07.

This page applies the [four steps from Before You Start](#) to hate speech and abusive language detection. It uses **AfriHate** as the example because it is the largest community-built hate speech collection for African languages, with native-speaker annotators and a published paper. Many smaller, single-language datasets exist, and new ones appear often. If your language is not mentioned here, go back to [Step 1](#) and search.

This task is different from the others in this chapter in three ways. Annotators read harmful content for hours, so they need support from the first day. A classifier used for automated moderation can cause the same harm it is meant to reduce. And what counts as hate is contested, culturally and politically.

Step 1: What a search turns up

Search Hugging Face, Lanfrica, or the ACL Anthology for hate speech in a major African language and you will find many small datasets, each with its own labels and guidelines. The one that brings most of them together is:

- [AfriHate](#) (Muhammad et al., 2025): tweets in 15 languages. Algerian Arabic, Amharic, Hausa, Igbo, Kinyarwanda, Moroccan Arabic, Nigerian Pidgin, Oromo, Somali, Swahili, Tigrinya, Twi, isiXhosa, Yoruba, and isiZulu. Each tweet is labelled hate, abusive, or normal, and hateful tweets also carry a target (ethnicity, politics, gender, disability, religion, or other).

The data is on the [Hugging Face Hub](#). The [GitHub repository](#) lists the language coordinators, who are good people to contact.

You will also find datasets for **nearby tasks** that look useful: emotion data such as [BRIGHTER](#) (28 languages, with labels that include anger and disgust), or polarization data such as [POLAR](#) (22 languages, including Amharic, Hausa, and Swahili).

Step 2: Judging AfriHate against the checklist

- **Licence.** The Hugging Face card lists Apache 2.0, which allows commercial use. The text is tweets, so also check whether the platform's terms affect what you plan to do.
- **Access.** Gated. You must log in to Hugging Face and agree to share your contact details before you can download it.
- **Fit.** Twitter/X posts from 2012 to 2023. Comments on other platforms, radio call-ins, WhatsApp messages, or forums look different. Hate speech also changes fast: new slurs, codewords, and spellings appear all the time, so older data misses them.
- **Script.** Mostly Latin script. Amharic and Tigrinya are in Ethiopic script, Algerian Arabic is in Arabic script, and Moroccan Arabic appears in both Arabic and Latin script.
- **Code-switching.** African-language hate speech often mixes in English, French, or Arabic, and the offensive part is often exactly where the switch happens. Check how much of your target text mixes languages, and include such text in your evaluation.

- **Labels.** Hate, abusive, and normal, plus a target for hateful tweets. What counts as hate is specific to a culture, a region, and a time: a word that is neutral in one place can carry a history of violence in another. Read the paper's definitions before assuming they match yours. Do not import a label scheme from an English corpus without consulting the affected communities.
- **Splits.** Fixed splits. Each test set has at least 100 examples of each class, so the rare hate class can be measured. Real content is mostly not hate, so also report results per class with the number of examples, not just one overall score.
- **Guidelines.** The paper describes the definitions, the keyword lists, and the annotation process. Check what has been released before planning to reuse them, and ask the authors if you cannot find the full guidelines.
- **Quality.** Annotated by native speakers from the local communities, with three or four annotators per tweet. Agreement is reported per language and ranges from 0.46 to 0.81 (Randolph's kappa), so quality varies by language. Look at the number for yours. The individual annotations are released, which helps if you want to study disagreement.
- **Documentation.** The paper has limitations and ethics sections. Tweets were found with keyword lists, so the data has selection bias: it only contains hate that matches the words the team searched for. The Twitter/X academic API used to collect it is no longer free, so extending the data the same way is harder today.

Step 3: Reuse, extend, or build

Find the row that matches your situation, then follow it across. The first question is whether your language is covered by AfriHate.

What you found	Your choice	What to do
Your language is covered by AfriHate, and its labels (hate, abusive, normal, plus target) and domain (Twitter/X) fit your task	Reuse AfriHate	Do not build a new corpus for these labels. Spend your effort on a small evaluation set from the platform you actually care about.
Your language is covered , but you need other labels or another platform	Extend it	For example, split incitement from slurs, or add calls to violence. Read the AfriHate paper first, extend its definitions rather than replace them, and involve the affected communities in label design, not just annotation. Do not re-annotate hate, abusive, and normal; merge.
Your language is not covered , but it is related to one of the 15	Borrow, then build small	Try cross-language transfer from the closest covered language, but treat it as a pilot only. What counts as hate does not transfer well between cultures. Build a target-language evaluation set of 500–2,000 examples before trusting any result.
Your language is not covered and not related to any of the 15	Build from scratch	Start with the long-tail language onboarding chapter, then Data Collection , Annotation Design , and Data Quality . Involve the affected communities in label design from the start, and aim for 5,000–10,000 examples. Do not start without counselling support in place for annotators.

Step 4: Learning from the people who built it

AfriHate was built by language teams across Africa, each with a coordinator, working under a shared set of labels. Its hardest problems were finding hateful posts in languages the Twitter API did not support, and handling honest disagreement about what counts as hate.

Read first:

- The [AfriHate paper](#) (Muhammad et al., 2025): the 15-language dataset, especially the data collection, limitations, and ethics sections.
- The [participatory research paper](#) (Nekoto et al., 2020): how to involve communities in dataset work. This matters more for label design here than for most tasks.

Questions to ask the authors:

1. How did you build the keyword lists, and what hate did they miss?
2. Where did annotators disagree most, and did disagreement follow region, age, or politics?
3. What support did annotators need, and what would you add next time?

Further reading

- [Challenges and frontiers in abusive content detection \(Vidgen et al., 2019\)](#): a survey of the method problems in this field. Read it before writing guidelines.
- [The risk of racial bias in hate speech detection \(Sap et al., 2019\)](#): how annotator background creates biased labels.

Once you have your data, training a model is well covered elsewhere. The [Hugging Face sequence classification tutorial](#) is a good start.

 [Cite this page](#)  6 min read

CONTRIBUTORS  (<https://github.com/abumafri>)  (<https://github.com/shmuhammadd>)

Dataset search log

Use this log while you work through [Before You Start](#). Fill in one copy per project. Keep it with your project files, and share it with your team. It shows that you checked for existing work, and it gives you the related-work section of your paper.



Download this template as a Word document (.docx), fill it in on your computer, or print it and fill it in by hand.

Project

Field	Your answer
Project name	
Task	e.g. sentiment analysis
Language	e.g. Swahili
Other names for the language	e.g. Kiswahili
All language codes	e.g. <code>sw</code> , <code>swa</code> , <code>swh</code>
Variety, script, and domain you need	e.g. Kenyan Swahili, Latin script, social media
Searched by	
Date started	

Step 1: Searches

Record every search, including those that found nothing. A search with no results is still useful evidence.

Build your search terms from three parts, and try several combinations:

- **Language:** every name and code, for example `Hausa`, `Harshen Hausa`, `hau`, `ha`.
- **Task:** the task name and its synonyms, for example `sentiment`, `opinion`, `polarity`, `emotion`.
- **Resource word:** `dataset`, `corpus`, `benchmark`, `annotated`, `shared task`.

The table shows example searches for a Hausa sentiment project. Replace them with your own.

Date	Where	Exact search terms	Relevant results
	ACL Anthology	e.g. <code>Hausa sentiment</code> , <code>Hausa corpus</code> , <code>Hausa dataset</code>	
	Lanfrica	e.g. <code>Hausa</code> , then filter by task	
	AtlasNLP	e.g. language: Hausa; task: sentiment analysis	
	Hugging Face (language filter)	e.g. language <code>hau</code> , task <code>text-classification</code>	
	AfricaNLP workshop proceedings	e.g. skim the last two years' paper titles for "Hausa" and "sentiment"	
	Google Scholar / arXiv	e.g. <code>"Hausa" sentiment dataset 2020..2026</code>	
	Asked the community (where?)	e.g. Masakhane Slack: "Does anyone know of Hausa sentiment data?"	

Step 2: Datasets found

One row per dataset. For each check, write **Pass**, **Partly**, or **Fail**, and add a short note where needed.

Dataset	Link	Languages	Size	Licence	Access	Fit	Labels	Splits	Guidelines	Quality	Documentation

Step 3: Decision

Question	Your answer
Which row of the Step 3 table applies?	Reuse / Extend / Build a new one, learning from the old one / Borrow, then build small / Build from scratch
Which dataset(s) is the decision based on?	
Why? (two or three sentences)	
Decided by, and date	

Step 4: People contacted

Date	Who (name, dataset)	How (email, Slack, meeting)	What you asked	What you learned

 [Cite this page](#)  2 min read

CONTRIBUTOR  (<https://github.com/shmuhammadd>)

Project Management

An African-language dataset is usually built by a small, distributed team: researchers, native-speaker annotators, and volunteers working part-time across several countries, time zones, and languages, often on a small grant. The Masakhane machine translation project, for example, was written by 49 authors across Africa and beyond, who coordinated almost entirely online ([Nekoto et al., 2020](#)).

Projects like these fail for reasons that have little to do with the data itself. In a study of 53 AI practitioners in India, East and West Africa, and the United States, 92% reported at least one "data cascade": a problem introduced early, in how data was scoped, collected, or labelled, that stayed hidden until it caused expensive damage later ([Sambasivan et al., 2021](#)). Many of these problems start with planning decisions. This chapter walks through those decisions in the order you will face them.

1. Scope the project

Mistakes made before collection are the most expensive, because every later step inherits them.

Check what already exists. Before you plan a new dataset, follow the four steps in [Before You Start](#). You may be able to reuse or extend an existing dataset, which changes the whole plan.

Write down the goal and how you will know you are finished. "A 5,000-sentence Yorùbá sentiment dataset, three annotators per item, inter-annotator agreement above 0.6, released under CC BY 4.0 by December" is a goal you can plan against. "Improve Yorùbá NLP" is not. A concrete goal sets the scale, the budget, and the stopping point, and it stops the project from growing until it runs out of money or volunteers.

Name the language precisely: the language, script, regional variety, and register. Swahili collected from Tanzanian news is not interchangeable with Kenyan social-media Swahili, and a model trained on one will underperform on the other. One language done well is usually better than three done badly. Multi-language projects are possible, but they are coordination problems first: MasakhaNER 2 reached 20 languages by running each language as its own sub-team under a shared protocol ([Adelani et al., 2022](#)).

Decide scale, modality, and task together, because they trade off against each other: how much data, in what form (text, speech, or images), and with what labels. Size the project to the team and budget you have. A small, clean, well-documented dataset is worth more than a large, noisy one, and quality problems are worst in large web-crawled collections for low-resource languages ([Kreutzer et al., 2022](#)).

Record these decisions in a project charter, and share it with the whole team and with your funder.



TEMPLATE: PROJECT CHARTER

A fill-in document that records the project's purpose, partners, language, data, licence, team, timeline, budget, governance, and ethics, with a signature block for every partner. [Open the project charter template.](#)

2. Settle ethics, consent, and the law early

These steps take longer than most teams expect, and they must be finished before you collect any data. Consent cannot be added to data you have already collected.

- **Ethics approval.** If you work at a university, you will usually need approval from a research ethics committee before collecting data from people. Review can take weeks or months, so apply as soon as the scope is fixed.
- **Data protection law.** Many African countries now have data protection laws, including South Africa (POPIA), Kenya (Data Protection Act, 2019), and Nigeria (Nigeria Data Protection Act, 2023). Check the rules in every country where you collect data or store it, especially if data will cross borders.
- **Consent and licence.** Decide how you will ask for consent and which licence you will release under before collection starts. The licence you want at the end limits what data you can collect at the start.

The [Legal, consent, and community IP](#) and [Data Governance](#) chapters cover these in detail.

3. Plan the work

Build the timeline backwards from your release date, through the stages that must happen in order: guidelines, pilot, revision, main annotation, quality control, documentation, and release.

Run a pilot first. Annotate 50 to 200 items with your real annotators and your real guidelines before you commit the full budget. A pilot shows you unclear instructions, hard cases, and disagreements while they are still cheap to fix. It also gives you a measured annotation rate (items per hour) to plan the rest of the schedule from.

Make a go/no-go decision after the pilot. Agree on the criteria beforehand, for example:

- Is agreement between annotators close to your target?
- Did the guidelines answer most of the annotators' questions?
- At the measured rate, does the full dataset fit the budget and the timeline?

If the answer to any of these is no, revise the guidelines or the scope and pilot again. Do not scale up a design that is not working.

Respect the order of dependent steps. Guidelines must be stable before main annotation starts, or you will re-annotate. Ethics approval and consent must be in place before collection.

Plan around the calendar. Many annotators are students or have other jobs. Exams, public and religious holidays, fasting periods, elections, and rainy seasons all change who is available and when. Put these dates in the plan and add slack.

4. Budget and fund the work

Estimate annotation cost first, because it is usually the largest cost for text projects:

$$\text{number of items} \times \text{annotators per item} \times \text{time per item} \times \text{hourly rate}$$

Then add time for review and adjudication, and for coordination. For example, 5,000 items with three annotators at one minute per item is 250 annotator-hours before any review. Use the rate you measured in the pilot, not a guess. The [Cost and Resource Planning](#) page covers estimation in more detail.

Budget for the costs that are easy to forget:

- Mobile data and airtime for annotators who work on their own phones.
- Devices, recording equipment, and storage for speech and image projects.
- Payment transaction fees, especially for cross-border payments.
- Coordination time for the project lead and language leads.
- A contingency line for re-annotation and for annotators who leave.

Pay annotators fairly and on time. Annotators are skilled contributors. Set a transparent rate based on local professional rates rather than the lowest global crowdsourcing rates, and agree on it before work starts. Mobile money (M-Pesa, MTN MoMo, Airtel Money) is often the most practical way to pay: it reaches people without bank accounts and arrives quickly. Late or unclear payment is the fastest way to lose a good team and to damage trust for the next project.

Plan for how money actually moves:

- Grant money often passes through a university or organisation, and payments can take weeks or months to reach annotators. Agree on a payment process with your finance office before work starts.
- Exchange rates change. If the grant is in dollars or euros but you pay in local currency, check the budget against the current rate before each payment round.
- Keep a record of every payment. Funders will ask for it, and it protects you and the annotators.
- Check the tax rules that apply to payments in each country.

Look for funding that targets dataset creation. [Lacuna Fund](#), founded in 2020 by The Rockefeller Foundation, Google.org, and Canada's IDRC, funds machine learning datasets in low-resource settings, including African languages. Other options include AI4D Africa, university partnerships, and in-kind support such as compute or annotator time. Include your data management and consent plan in the proposal. Funders increasingly expect it.

5. Build the team

Most projects have more roles than people, so one person often holds several roles. What matters is that every role has a named owner.

Role	Responsible for
Project lead	Schedule, budget, communication, and reporting to the funder.
Language lead (one per language)	Guidelines, hard cases, and quality for that language. The final authority on what is correct.
Technical lead	Annotation tool, data pipeline, storage, and backups.
Ethics and data lead	Ethics approval, consent records, data protection, and licensing.
Annotators	Producing labels.
Reviewers	Checking quality and resolving disagreements.

Keep annotation and review separate, even if people rotate between them. Reviewing your own work hides the errors you most need to catch. A language lead for each language, rather than one central team labelling everything, is what let Masakhane projects work across many languages at once ([Nekoto et al., 2020](#)).

Recruit early. Finding fluent native speakers is often the hardest constraint, especially for smaller languages. Give recruitment its own time in the plan. Good places to look include university language and linguistics departments, community organisations, and existing research communities such as [Masakhane](#) [↗]. Ask candidates to label a short test set before you hire them, and train everyone on the guidelines before the pilot.

Put agreements in writing. Each contributor should have a short written agreement that covers pay, working hours, how their data and work will be used, the licence, and how they will be credited.

Agree on credit at the start. Decide early who will be an author on papers, who will be acknowledged, and how annotators will be named in the dataset documentation. Clear, fair credit keeps volunteers involved, and it avoids disputes at release.

6. Coordinate and track

Work asynchronously. Distributed, part-time teams need simple tools that work on a phone and on a weak connection. For most projects, three things are enough:

- one shared task tracker (a spreadsheet works);
- one place for the current version of the guidelines;
- one agreed channel for questions, such as a WhatsApp or Slack group.

Write decisions down, because few people are online at the same time.

Review a few numbers every week or two:

- items completed against the plan;
- agreement between annotators;
- how many annotators are active;
- cost per item so far.

Falling agreement usually means the guidelines need fixing, not that annotators need correcting. Annotators dropping out usually points to problems with pay, workload, or distressing content. A problem caught in week two costs far less than one found at release ([Sambasivan et al., 2021](#)).

7. Manage risks

These risks come up in almost every African-language data project:

Risk	Early warning sign	What to do
Annotators leave	Fewer items per week; slow replies	Set realistic timelines, pay on time, and recruit a slightly larger pool than you need.
Guidelines drift	Agreement falls over time	Keep the guidelines versioned, and re-check annotators on shared items regularly.
Payments are delayed	Finance office has not confirmed the process	Agree on the payment process before work starts, and keep a small float for urgent payments if your organisation allows it.
Currency changes	Local currency falls against the grant currency	Recheck the budget before each payment round, and keep a contingency line.
Connectivity and power fail	Annotators miss deadlines in the same region	Choose tools that work offline or on a weak connection, and cover mobile data costs. Government internet shutdowns have happened in several countries, so plan for gaps.
Funding or a tool disappears	Grant end date approaches; tool changes its pricing	Keep your own copies of all raw data and exports, and avoid depending on a single service.
Harm to annotators	Annotators skip items or leave after sensitive batches	Warn annotators about sensitive content, let them opt out, limit exposure, and review data before release.

8. Release and hand over

Plan the release from the start. Decide where the dataset will be hosted, under which licence, how versions will be numbered, and who will answer questions and fix errors after the grant ends. The [Documentation](#) chapter explains how to write a datasheet for the release.

Assume someone else will finish the project. Team members graduate, change jobs, or run out of time. Keep the guidelines, schema, consent records, and processing scripts in a shared repository, not on one person's laptop. This lets the next person pick up the work, and it lets you show where the data came from years later.

Write down what you learned. When the project ends, record what worked and what did not, using the [retrospective template](#). It helps the next team plan better than you could.

Checklist before you start

- ☐ Searched for existing datasets ([Before You Start](#)).
- ☐ Goal, language variety, scale, and success criteria written in a project charter.

- ☐ Ethics approval and data protection requirements identified, with time in the plan.
- ☐ Consent process and release licence decided.
- ☐ Pilot planned, with go/no-go criteria agreed.
- ☐ Budget built from a measured rate, including hidden costs and contingency.
- ☐ Payment process agreed with your finance office.
- ☐ Every role has a named owner.
- ☐ Written agreements and a credit policy for all contributors.
- ☐ Release, hosting, and maintenance plan in place.

HOW THIS CONNECTS

The decisions in this chapter set up [Data Collection](#), [Annotation Design](#), and [Data Quality](#), which follow.

 [Cite this page](#)  10 min read

CONTRIBUTORS



(<https://github.com/abumafrim>)



(<https://github.com/shmuhmadd>)

Project charter template

Last reviewed: 2026-07-07.

The Step-0 community agreement covering orthography, use case, IP framework, and team structure that the [long-tail language onboarding](#) chapter argues must be settled in writing before data collection begins. Fork it, negotiate it with the community, sign it, and store it alongside the corpus release.

How to use this template

1. Copy everything below the divider into a new file ([docs/project-charter.md](#)) in your project repo or the community-owned repository).
2. Fill in every `[BRACKETED FIELD]` with a real answer negotiated with the community, not decided by the project team alone.
3. Convene the community consultation described in [Section C of the consent template](#). The charter is the artefact that comes out of that consultation.
4. Sign it. All named parties (project lead, community stewards, institutional sponsor, ombudsperson) sign a physical or digitally-witnessed copy. Store one copy in each party's possession.
5. Version it. The charter can be amended; amendments require the same signature process.

Motivated by [long-tail language onboarding Step 0](#), the [legal, consent, and community IP](#) chapter, and every Case Study in the playbook where a missing charter shows up as the root cause of a downstream dispute.

Project charter: [PROJECT NAME]

Version: [X.Y] · **Effective date:** [YYYY-MM-DD] · **Language(s):** [LANGUAGE(S)] · **Community(ies):** [NAMED COMMUNITIES]

1. Purpose

- **What this project sets out to build:** [One sentence in plain language, not the funder-facing version.]
- **The problem this is trying to solve for the community:** [The community's articulation of the problem, not the project team's. If these are different, both are named here.]
- **What success looks like at the end:** [Specific enough that a reader can tell if it happened.]
- **What is explicitly out of scope:** [Preventive; naming what will NOT be built stops scope creep later.]

2. The parties

- **Project lead:** [NAME + affiliation + contact + role.]
- **Institutional sponsor:** [ORGANISATION legally responsible for the project + point-of-contact.]
- **Community stewards:** [NAMED community representatives with the standing to speak for the community on this project.]
- **Ombudsperson:** [NAME + contact; see the [ombudsperson role](#) in the consent template.]

- **Technical lead:** [NAME + role.]
- **Community liaison:** [NAME + role, distinct from the annotator role.]
- **Linguistic consultant, if any:** [NAME + role.]
- **Funders:** [Grantors and their conditions; if a funder requires acknowledgement or specific outputs, name them here.]

3. Language and orthography

- **The language(s) the corpus covers:** [Specific varieties, dialects, registers included.]
- **Varieties, dialects, registers explicitly excluded:** [What is not in scope.]
- **Script(s) used:** [Which script(s), with the community-negotiated reasoning.]
- **Orthographic convention:** [Which standard is followed; if the community is genuinely split, whether the project ships dual conventions or picks one, and how the choice was made.]
- **Diacritic and tone convention:** [The specific rules that apply. Reference for annotators.]
- **How orthographic decisions can be revisited:** [Which body has standing to revise, and by what process. Orthographic conventions drift; the charter names how the corpus will handle drift.]

4. Data: what is collected, from whom, how

- **Types of data collected:** [Speech, text, image, video, annotations, translations. Be specific.]
- **Sources of data:** [Community-produced original content, existing texts, elicited translations. Ranked and justified. See the [long-tail chapter's source-selection ranking](#).]
- **Contributors:** [Who contributes: recording speakers, translators, annotators. Anonymised or named-with-consent.]
- **Consent framework:** [Reference to the [consent form template](#) or the project's specific consent artefact.]
- **Storage:** [Location (country + institution + system), access controls, retention period.]

5. Community IP and licence

The centre of the charter. Every question below is answered explicitly.

- **Who owns the raw data?** [Contributor, community, project, with the reasoning.]
- **Who owns the annotated corpus?** [Named party or shared-ownership arrangement.]
- **Who owns the models built from the corpus?** [Same.]
- **Under what licence is the corpus released?** [SPDX identifier + reasoning. See the [licence selection section](#). CC BY-NC 4.0 is the playbook's honest default; document why if choosing something different.]
- **Under what licence are models released?** [Model release is a separate decision from data release; document it.]
- **What re-use requires community approval?** [E.g., "commercial use", "inclusion in a downstream release under a different licence", "use in a foreign-owned model's training data".]
- **Who has the standing to approve re-use requests?** [Named body or steward.]
- **How is benefit-sharing structured?** [If the corpus generates downstream revenue, how does that flow back to the community? If it doesn't, be honest that it doesn't.]

6. Team, timeline, and budget

- **Timeline:** [Milestone dates. See the [long-tail chapter's milestone-0-to-4 arc](#) for reference; use a version realistic to this project.]
- **Effort estimate:** [Person-months per role.]
- **Budget summary:** [Total budget + high-level breakdown. Detailed cost accounting can live in a separate financial document.]
- **Compensation policy:** [Rate for annotators, translators, community liaison, ombudsperson. Rates are named to prevent the failure mode of paying different contributors differently for the same work.]
- **What happens if the timeline slips:** [Renegotiation process; scope decisions.]
- **What happens if the budget is exhausted:** [Priority order for what gets finished vs. cut.]

7. Governance and decision-making

- **Who makes technical decisions:** [Named person or role.]
- **Who makes community-facing decisions:** [Named person or role, usually a community steward, not the project lead.]
- **How disputes are resolved:** [Escalation path; ombudsperson role; final arbiter.]
- **How the charter is amended:** [Amendment process; who has to agree; documentation requirements.]
- **How the project can be paused or halted:** [The ombudsperson's authority; the community's authority; the sponsor's authority. State them.]

8. Deliverables and commitments

- **What the project will deliver:** [Corpus, models, documentation, publications, deployment. Be specific.]
- **What the project will NOT deliver:** [Managed expectations. Any deliverable not on the will-list is out of scope; adding one requires a charter amendment.]
- **Publication policy:** [Who is credited; where results are published; what the community's role in publication approval is.]
- **Post-project maintenance:** [Who owns the corpus after the project ends; who fields re-use requests; what the community's ongoing rights are.]

9. Ethics and safeguards

- **Ethics review:** [IRB or ethics-review-board name + date + outcome. If no formal ethics review is possible, describe the substitute community-review process.]
- **Data protection compliance:** [Which country's law applies; how compliance is documented. See the [country reference](#).]
- **Vulnerable-population protections:** [If the project touches specific vulnerable populations (children, refugees, minority-within-the-community groups), the additional protections in place.]
- **Cultural-heritage protections:** [If the corpus includes traditional knowledge or culturally-sensitive material, the protections around it. See [WIPO's traditional knowledge framework](#) ↗.]

10. Signature block

By signing below, each named party confirms they have read this charter, agreed to it, and understand their responsibilities under it.

Project lead: _____ Date: _____

Institutional sponsor: _____ Date: _____

Community steward(s): _____ Date: _____

_____ Date: _____

Ombudsperson: _____ Date: _____

Independent witness: _____ Date: _____

11. Amendments

Amendment	Date	Change	Approved by
1	[DATE]	[SHORT DESCRIPTION]	[SIGNATORIES]

Contributor's note. If you have negotiated a working charter for an African-language NLP project, the highest-value contribution is a redlined comparison against this template with the terms you actually agreed to. Real charters from real projects reveal what fields the template got wrong and where community-specific additions belong.

 [Cite this page](#) ⌚ 6 min read

CONTRIBUTORS  (<https://github.com/abumafri>)  (<https://github.com/shmuhammadd>)

Data Governance

Data governance is the set of decisions about who controls a dataset: who may access it, who may use it, who profits from it, and who is accountable when something goes wrong. For African-language data these are not legal formalities to add at the end. They decide whether a dataset strengthens a community or quietly extracts from it. The data comes from people, from their voices, words, and knowledge, and governance is how those people keep a say in what they helped create.

This matters more here than in well-resourced settings, because the same openness that helps a low-resource language can also expose it. A freely downloadable corpus is easy to reuse, and just as easy to scrape into a commercial model that returns no benefit, credit, or say to the speakers. African NLP communities have been clear about this. The Masakhane community holds that Africans should decide what data represents their communities, retain ownership of it, and know how it is used ([Masakhane](#)). Governance is how that principle turns into practice.



THE ONE-LINE VERSION

Decide with the community who controls the data, get real consent, choose a licence on purpose, protect the people in the data, and make sure the benefits reach the speakers.

FAIR and CARE, together

Two frameworks that answer different questions — use both.

FAIR

makes data usable

Findable

Data can be located and indexed.

F

Accessible

It can be retrieved under clear terms.

A

Interoperable

Works with other data and tools.

I

Reusable

Documented well enough to use again.

R



CARE

makes data just

Collective benefit

Value returns to the community.

C

Authority to control

The community decides what is shared.

A

Responsibility

Serve the community, not just study it.

R

Ethics

Rights and dignity guide every choice.

E

FAIR without CARE is well-documented extraction.

Data can be perfectly FAIR and still be taken from a community that gets nothing back.

Govern for the community, not just the dataset

Community ownership and self-determination

The people whose language it is should have the final say over how their data is used. This participatory commitment, treating native speakers as co-owners rather than suppliers of raw material, runs through the continent's strongest data efforts. It shaped the early Masakhane translation work ([Nekoto et al., 2020](#)), AI4D's African Language Program, which built open datasets through community challenges and short research fellowships ([Siminyu et al., 2021](#)), and large speech corpora such as NaijaVoices that are recorded by community members rather than scraped ([Emezue et al., 2025](#)). It also reflects a wider call to decolonise language technology by centring the communities whose languages are at stake instead of treating them as data sources ([Bird, 2020](#)). Self-determination is not a single veto at release. It shapes every earlier choice: what gets collected, from whom, on what terms, and who may use the result.

FAIR and CARE, together

Two frameworks should guide a dataset's design, and they answer different questions. FAIR asks whether data is Findable, Accessible, Interoperable, and Reusable, and it is now a baseline expectation for research data ([Wilkinson et al., 2016](#)). FAIR says nothing about power or justice, though. Data can be perfectly FAIR and still be taken from a community that gets nothing back. CARE was written to fill that gap. Its four commitments, Collective benefit, Authority to control, Responsibility, and Ethics, assert the right of Indigenous and local communities to govern data about them ([Carroll et al., 2020](#)). Use both. FAIR makes the data usable; CARE makes it just. FAIR without CARE is well-documented extraction.

The 'CARE Principles for Indigenous Data Governance' (Collective Benefit, Authority to Control, Responsibility, and Ethics) empower Indigenous Peoples by shifting the focus from regulated consultation to value-based relationships that position data approaches within Indigenous cultures and knowledge systems to the benefit of Indigenous Peoples. This shift ultimately promotes equitable participation in processes of data reuse, which will result in more equitable outcomes.

Carroll et al. (2020)

CARE was written for Indigenous Peoples' data, and it asks three questions of any dataset: whom the data serves, who benefits from it, and who has authority over its use ([Imperial College London](#) gives a short summary). An African-language project can ask the same questions on behalf of the community whose language it records. Each principle maps to a decision in this chapter:

Principle	What it asks	In your project
Collective benefit	The data should benefit the community it comes from, through development, better governance, and fairer outcomes.	Agree what returns to the speakers before you collect. See Benefit-sharing and credit .
Authority to control	The community's rights and interests in its data are recognised, and the community governs how the data is used.	Write down who decides, and keep that authority with the community. See Governance models .
Responsibility	People who work with the data build respectful relationships, grow local skills, and respect the community's language and worldview.	Hire and train locally, and tell the community how its data is being used.
Ethics	The community's rights and wellbeing come first at every stage of the data's life, including uses nobody has planned yet.	Review possible harms before collection and again before each release. See A lightweight ethics review .

A community can also state its own conditions on a dataset. [Local Contexts](#) provides Traditional Knowledge and Biocultural labels for this purpose, which travel with the data and tell a reuser what the community permits.

The findability half of FAIR has a practical home in African NLP. Lanfrica's African AI Atlas tracks datasets, models, and papers that would otherwise sit scattered across repositories, PDFs, and dead project pages, so that work already done can be found and reused instead of rebuilt from scratch ([Lanfrica](#)).

Consent and rights

Informed consent

Anyone who contributes data, whether a voice recording, a translation, an annotation, or a photo, should know before they contribute what they are agreeing to: what the data is for, who will be able to use it, whether it will be public, and that they may decline or withdraw without penalty. This is not optional for the recording-based collection common in African-language work, where the contributor's own voice or image is the data. NaijaVoices shows what good practice looks like. To build an 1,800-hour Igbo, Hausa, and Yorùbá speech corpus from more than 5,000 voice donors, it trained community facilitators in the ethics of data collection and informed consent, and those facilitators explained the project, its purpose, and the intended use of the data to each contributor before any recording began ([Emezue et al., 2025](#)). Gather informed consent, fair compensation, and clear data-rights terms together, at the point of collection ([Esethu Framework, 2025](#)). Adding consent afterward is usually impossible, which is what makes this an early decision rather than a late one.

Rights over sourced and contributed data

Be clear about which kind of data you hold, because the rights differ sharply. Contributed data, the recordings, transcriptions, and labels people make for your project, is governed by the consent and terms you agree with them. Sourced data, the text pulled from the web, social media, or archives, carries someone else's rights: copyright, platform terms of service, and often no clear licence at all. African NLP practitioners have flagged copyright and access as a central, unresolved problem, since much of the available text sits in a legal grey zone and "publicly visible" is not the same as "free to redistribute or train on" ([Carnegie Endowment, 2024](#)). When the rights are unclear, record the uncertainty rather than hide it.

Licensing

A licence tells the world what may and may not be done with your data. In African NLP the common failure is not the wrong licence but a missing or unclear one, and an unlicensed dataset is effectively unusable, because a reuser cannot tell whether they are allowed to touch it ([Lanfrica](#), "[Licensing as a Barrier](#)"). Choose on purpose.

Data and code are licensed differently

Software licences such as MIT or Apache-2.0 are not built for data, and putting a code licence on a corpus leaves its reuse rights ambiguous. For open data the Creative Commons family is the usual choice: CC0 for a public-domain dedication, CC BY for reuse with attribution, or CC BY-SA for attribution plus share-alike. Pick the one that matches how open you actually want to be, and apply it to the data, separately from any code.

African and community licences

Fully open CC licences assume the goal is maximum reuse. For low-resource African data, maximum reuse often means a well-funded foreign lab can take the corpus, train on it, and give nothing back to the speakers. A newer class of community-centred licences exists to close that gap, and the most developed of them is African.

The Nwulite Obodo Open Data License (NOODL) takes its name from the Igbo for "raising, reviving, and building the community," and it was written to address the inequitable openness that standard licences impose on African dataset creators ([NOODL, 2025](#)). Its central idea is to set terms by who the reuser is. Users in Africa and other developing nations may reuse the data freely on a share-alike basis and distribute adaptations within the region. Users outside those regions accept the same share-alike terms and must also return royalties or other benefits to the African dataset owners. Openness becomes reciprocal rather than a one-way export.

The Esethu license, from Lelapa AI, Way With Words, and Data Science for Social Impact, adds an economic loop: licensing revenue is reinvested into expanding the dataset and supporting local employment, so the resource and its community grow together ([Esethu Framework, 2025](#)). Both build on the precedent of Te Hiku Media's Kaitiakitanga License for Māori data, which treats data as cared for rather than owned, directs benefit back to the source community, and forbids uses such as surveillance or unconsented corpus-building ([Te Hiku Media](#)). Consider one of these whenever a plain CC licence would mean giving up the community's say, or its share of the value its data creates.

Attribution and reuse

Whatever licence you pick, require attribution and record provenance: where the data came from, who contributed it, and on what terms. Provenance lets downstream users credit the source and respect the conditions, and it lets you prove years later that the dataset was built legitimately. Recording it is covered in [Documentation](#).

Privacy and sensitive data

Personally identifiable information

African-language collection routinely captures personal data: names and places in text, and, unavoidably, voices in audio and faces in images, which identify a person directly. Collect as little as the task needs, anonymise where you can by masking names and stripping metadata, and never treat a recorded voice as anonymous. Where identifiable data is essential, it has to rest on explicit, informed consent, and the consent terms must travel with the data.

Sensitive and restricted content

Some knowledge should not be public even when it is easy to collect. Sacred or ceremonial material, culturally restricted knowledge, health information, and politically dangerous speech can all put contributors at real risk. This is what CARE's Authority to control protects: the community, not the collector, decides what may be shared and what stays closed ([Carroll et al., 2020](#)). When unsure, ask the community and default to not releasing.

The law that applies: African data-protection regimes

Across the continent, governance is now law as much as ethics. By 2026, 44 African countries had enacted data-protection legislation, and most had a functioning authority to enforce it ([Tech In Africa, 2026](#)). If you collect personal data, and a recorded voice is personal data, you are almost certainly covered by one of these regimes:

- **South Africa, POPIA** (Protection of Personal Information Act, 2020): a rights-based law that calls for a lawful basis, purpose limitation, and an impact assessment for automated processing.
- **Nigeria, Data Protection Act (2023)**, building on the earlier NDPR: consent, data-subject rights, and registration of data controllers.
- **Kenya, Data Protection Act (2019)** and its 2021 regulations, alongside the 2025–2030 National AI Strategy, which adds dataset-level expectations including proposed audit trails for AI training data.
- **Ghana, Data Protection Act (2012)**: one of the continent's earliest.
- **African Union, Malabo Convention**: the first continent-wide treaty on personal-data protection and cybersecurity, in force since June 2023 ([Malabo Convention, 2023](#)).

The practical demands are similar across these laws: collect on a lawful basis, usually informed consent; use the data only for the purpose you stated; honour data-subject rights to access, correction, and deletion; and be ready to run an impact assessment for anything that feeds automated decisions. Check the law of every country your contributors are in, not only your own. African Next Voices makes the point concrete. It gathered roughly 9,000 hours of speech across Kenya, Nigeria, and South Africa ([African Next Voices, 2025](#)), which placed one project under three separate data-protection regimes at once.

Ethics and harm avoidance

A lightweight ethics review

Before collection starts, work through a short and honest review: who could be harmed by this dataset, how, and what would reduce the risk. It does not need to be a formal institutional board, which many community projects cannot access, but it does need to happen before data is gathered, while choices are still cheap to change. How the work is done, and who is consulted along the way, often decides whether a community is genuinely served or merely studied, which is what CARE's Responsibility and Ethics commitments ask you to attend to ([Carroll et al., 2020](#)).

Bias, representation, and harm

Ask who is represented in the data and who is missing, across dialects, regions, genders, and ages, because those gaps become the blind spots of any model trained on it. Tasks that involve harmful content, such as hate speech or offensive language, carry a second duty of care toward the annotators who have to read it, met through content warnings, opt-outs, and support (see the annotation chapters). Documenting known biases plainly is part of governance, not an admission of failure.

Community ownership and benefit-sharing

Governance models

Decide in writing who decides, and keep that authority with the community. In practice this can be a community advisory board, a designated language lead for each language, or a formal data-trust arrangement, and the right model depends on the project's scale and the community's structure. What matters is that the decision rights are explicit and held by the people whose language it is, rather than defaulting to whoever happens to host the files.

Benefit-sharing and credit

Governance is hollow if value only flows outward. Benefit-sharing means the gains from a dataset, whether recognition, compensation, capacity, tools, or a share of downstream value, return to the contributors and their community. This is the organising idea behind the Kaitiakitanga License's principle that benefit flows to the source, the Esethu framework's reinvestment of licensing revenue into the dataset and local jobs, and NaijaVoices' "data farming" model, which pairs collection with reciprocal community support and micro-grants for local language work ([Te Hiku Media](#); [Esethu Framework, 2025](#); [Emezue et al., 2025](#)). At a minimum, credit the annotators, speakers, and communities by name in datasheets and publications. They are co-creators, and naming them is the cheapest benefit-sharing there is.

HOW THIS CONNECTS

Governance decisions made here set the terms for everything downstream: what you may [collect](#), how you run [annotation](#), and what you record in [Documentation](#). Get governance right early and the rest of the project stands on solid ground.

 [Cite this page](#)  12 min read

CONTRIBUTORS



(<https://github.com/abumafrim>)



(<https://github.com/shmuhammadd>)

Governance Toolkit

The [Data Governance](#) and [Legal, consent, and community IP](#) chapters explain why a dataset needs a licence, agreements with the people who build it, and a plan for its life after the grant. This toolkit is the set of documents that puts those chapters into practice. Each one is a file you can copy, fill in, and sign.

What is in the toolkit

Tool	What it decides	Use it when
Licensing decision tree	Which licence, or which restricted-release pattern, your dataset can honestly carry, given the consent you collected and the sources you used	Before the project charter is signed, and again before release
Contributor agreement	The terms for volunteers and collaborators whose work becomes part of the dataset: what they grant, how they are credited, how they can stop	When onboarding anyone who is not paid per task
Annotator contract	The terms for paid annotation, transcription, or translation work: pay, rejection, wellbeing, data protection, rights	Before the first paid task is assigned
Sustainability plan	Who stewards the dataset after the grant, what it costs, and where the money and time come from, including a business model for services on open data	At project design, and again in the last quarter of the grant

Three related documents live elsewhere and the toolkit assumes them: the [consent form](#), which governs what speakers and contributors agreed to and bounds every licensing choice; the [project charter](#), which records the community's decisions; and the [data ownership record](#), which names who decides.

How the pieces fit

The documents constrain each other, and the order matters.

1. **Consent first.** The consent form fixes what the data may be used for. No licence can grant more than the speakers agreed to.
2. **Then the licence.** The decision tree turns the consent terms, the source-data licences, and the community's wishes into one of a small set of outcomes.
3. **Then the agreements.** The contributor agreement and the annotator contract grant the project exactly the rights it needs to release under that licence, and no more. They also carry the project's promises back: credit, pay, and the right to withdraw.
4. **Then the plan.** The sustainability plan costs the steward's time, corrections, and hosting, and says who pays. Its business model section must respect the licence chosen in step 2.

The three experts who drafted these documents worked from different disciplines, intellectual property law, labour law and research ethics, and development economics, and reviewed each other's drafts until the rules agreed. Where the documents state a shared rule, such as how a project may or may not change its licence later, that rule is the same in every file.

What the toolkit is not

It is not legal advice. Every template carries a disclaimer for a reason: data protection, employment, and copyright law differ by country, and several of the questions the toolkit raises have no settled answer in any African jurisdiction. Use the documents to arrive at a well-reasoned draft, then have someone qualified in the relevant country read it.

It is also not a substitute for the conversation with the community. The templates carry opinionated defaults, following the playbook's [editorial rule for templates](#), so that a project starts from a good position rather than a blank page. The community's answer overrides the default every time.

 [Cite this page](#)  3 min read

CONTRIBUTOR  (<https://github.com/shmuhammadd>)

Licensing decision tree

A licence can only grant what the people in the data agreed to and what the sources you built on allow. This tree walks those two constraints first, then the community's wishes, and ends at one of five outcomes. Work through it before the [project charter](#) is signed, and again before release, because consent terms and source data often change during a project. The reasoning behind each licence is in [Data Governance](#); the practical notes on each type are in [Licensing](#).

! BEFORE YOU START

Have three things in front of you: the consent form your contributors signed or recorded, the licence of every dataset, corpus or model output you reused, and the community's decision on who may benefit from the data (recorded in the charter). The tree asks about each.

Work through the tree

Printed version of the tree.

1. Did every contributor consent to public release? No or unclear: do not release; re-consent or drop the items.
Research-only consent: go to restricted release (step 6).
2. Does the dataset include material you did not create? If unchecked: stop and list every source with its licence. If a source is share-alike, non-commercial or competition-only and cannot be removed: your licence inherits the strictest term (CC BY-SA, CC BY-NC, or remove the material).
3. What did the community decide about benefit? Anyone may reuse with credit: step 4. Benefit must flow back: step 5.
Named partners only: step 6. Some material must stay closed: step 7.
4. Should adaptations stay open under the same terms? Yes: CC BY-SA 4.0. No: CC BY 4.0.
5. Community licence: NOODL for a community-held corpus, Esethu when a steward will license the data and reinvest the revenue.
6. Restricted release: gated access with a data-use agreement; publish the datasheet, schema and a consented sample openly.
7. Split the release: keep the restricted partition closed and run the open remainder through the tree again.

After the tree

Write the outcome into three places: the licence field of the [dataset card](#), the licensing agreement in [Licensing](#), and the sustainability plan's business-model section, which must respect the licence you chose. Then check the two agreements that grant the project its rights: the [contributor agreement](#) and the [annotator contract](#) must grant at least what the licence gives away, and no more.

A licence is hard to change after release. Copies already downloaded keep the terms they were released under, so a later move from CC BY to a community licence only binds new downloads. If the community may want to tighten terms later, start with the tighter licence.

 [Cite this page](#)  2 min read

CONTRIBUTOR  (<https://github.com/shmuhammadd>)

Contributor Agreement Template

A contributor agreement sets out the terms for people who help build the dataset, such as annotators, translators, and chapter authors, covering what they will do, how their work is licensed, and how they are paid and credited. Adapt it to your project and local law.

When to use this template

When onboarding annotators, translators, or other contributors whose work becomes part of the dataset or the playbook.

What this agreement covers

The scope of the contribution, the rights granted, compensation and attribution, data handling, and how either side can end the arrangement.

Part 1. Contributor and project details: [contributor name, contact]; [project, coordinator].

Part 2. Scope of contribution: the contributor will [annotate / translate / write / review] [describe the work and volume] to the standards in the project guidelines.

Part 3. Intellectual property and licensing: the contributor grants the project the rights needed to include and release their contribution under the dataset's licence [licence], while [retaining / transferring] their own rights as agreed.

Part 4. Compensation and attribution: compensation is [amount, schedule, method, e.g. mobile money]; the contributor will be credited as [name / pseudonym / anonymous] in datasheets and publications.

Part 5. Confidentiality and data handling: the contributor will handle any sensitive or personal data only as the project directs and will not redistribute it.

Part 6. Termination: either party may end the agreement with [notice]; completed and paid work remains licensed as above.

Part 7. Signature: [names, dates, signatures].

Pay fairly and on time, and make attribution real. A contributor agreement is also a statement of respect for the people building the data (see [Project Management](#) and [Data Governance](#)).

 [Cite this page](#) ⌚ 2 min read

CONTRIBUTOR  (<https://github.com/shmuhammadd>)

Annotator contract template

Draft in progress.

 [Cite this page](#)  1 min read

CONTRIBUTOR  (<https://github.com/shmuhammadd>)

Sustainability Plan Template

A sustainability plan answers what happens to the dataset after the grant ends. Datasets that plan for this keep living; those that do not quietly die. Funders increasingly ask for one, and it is worth writing for your own sake regardless.

When to use this template

At project design, to think it through early, and again at the end of a grant, to put it into action.

What this plan covers

Who stewards the data after the funding ends, where it lives, how it is maintained and updated, and how the work is resourced.

Part 1. Project and dataset overview: [what the dataset is and who it serves].

Part 2. Stewardship and governance after the grant: decision rights and day-to-day stewardship pass to [named community body or institution].

Part 3. Hosting and access: the dataset will remain available at [stable host, e.g. Hugging Face, Zenodo, institutional] with [a stable identifier].

Part 4. Community maintenance and update process: errors and updates are handled through [issue tracker / forum], with [who] responsible for responding.

Part 5. Translation and localization pipeline: if the resource is to grow to new languages or varieties, the process for that is [described here].

Part 6. Funding and resource strategy beyond the grant: ongoing costs are met by [institutional support / follow-on funding / in-kind community effort].

Part 7. Risk and contingency: key risks are [e.g. host shutdown, loss of key people], mitigated by [plans].

Part 8. Milestones and review schedule: the plan is reviewed [cadence].

Sustainability is the lifecycle question seen from the funding side. It connects directly to [Maintenance](#) and [Community](#).

 [Cite this page](#) ⌚ 2 min read

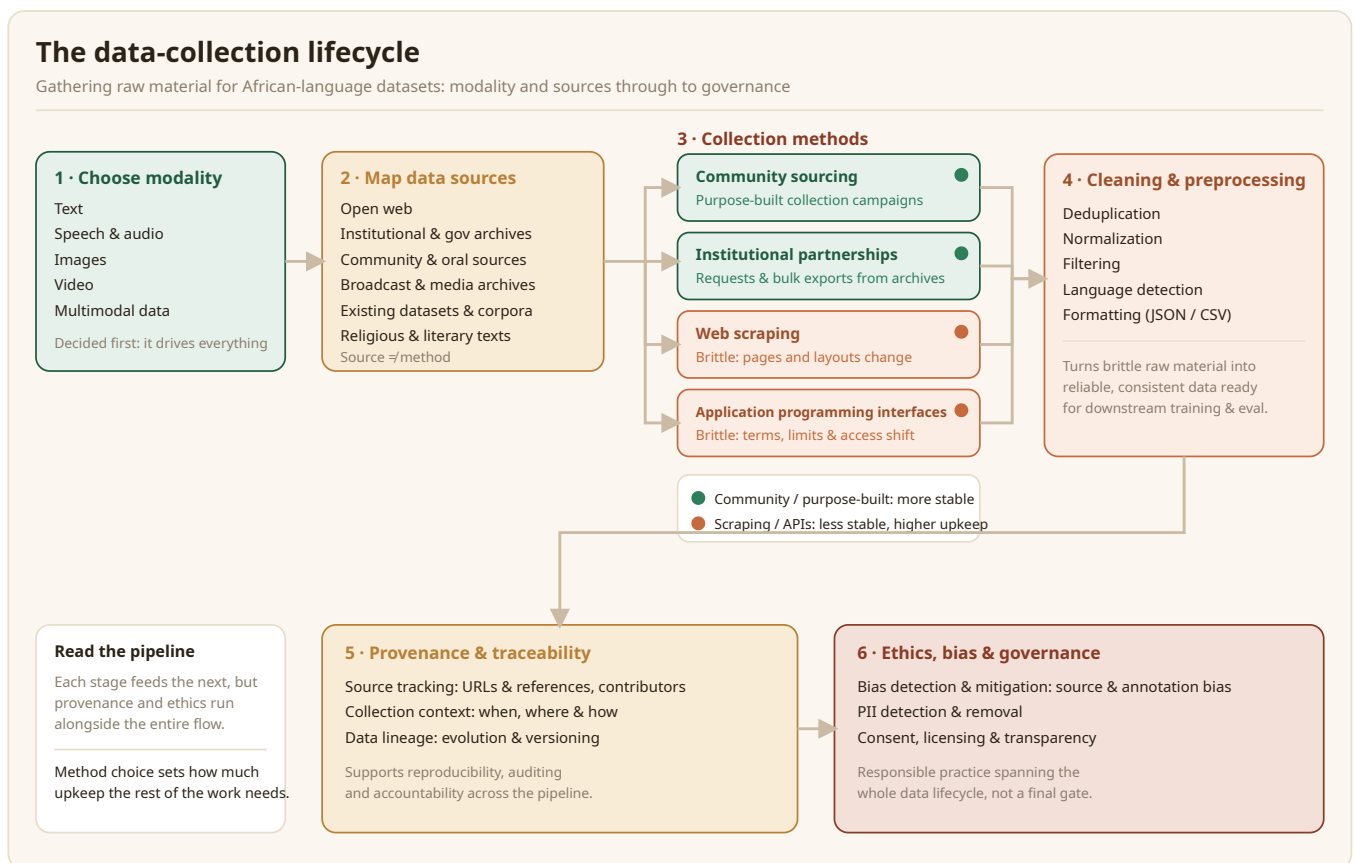
CONTRIBUTOR  (<https://github.com/shmuhammadd>)

Data Collection

Every dataset starts the same way: someone has to go and get the raw material. This chapter is about that first step, what the rest of the playbook calls **data collection** or **data curation**. We mean this narrowly: the actual gathering of raw text, images, audio, or video, before anyone has cleaned it, labelled it, or decided what it means. What happens after collection (cleaning, annotation, quality control, and release) is covered in the chapters that follow.

How you collect data depends almost entirely on its **modality**: the format the data takes, whether that's text, images, audio, or video. Each modality has its own sources, its own tools, and its own failure modes, and a method that works well for one (scraping news sites for text) can be close to useless for another (there is no equivalent "scrape the web" shortcut for spoken-language audio in a language with no written tradition).

The sections below work through the building blocks in order: what modality you're collecting, where it can come from, two of the most common collection mechanisms in detail (web scraping and APIs), and a worked case study showing how these decisions interact when a project spans more than one modality at once. The chapter closes with what happens once raw data is in hand: planning the resources it took to get there, cleaning it, tracking where it came from, and governing it responsibly.



[Cite this page](#) ⌚ 2 min read

CONTRIBUTORS



(<https://github.com/abumafrim>)



(<https://github.com/shmuhammadd>)

Data Modalities

Learn what data modality means, why it's the first decision in any collection plan, and how it shapes cost, sourcing, and tooling long before anyone writes an annotation guideline.

Data modalities you'll most often collect

The format your data takes is the first fork in the road: it shapes sourcing, cost, who can supply it, and what can go wrong.

Text

Sequences of discrete symbols: the most common start, with the most tooling.

Needs:

Web crawls, archives, books, social media, but apparent abundance is often deceptive.

Speech & audio

Often the more natural modality for oral-tradition languages with no set orthography.

Needs:

Recording, not scraping: equipment, consent, and transcription as its own annotation task.

Images

Dense grids of pixels. Two problems: source existing images, or capture new ones.

Needs:

Sourcing raises licensing questions; capture raises consent and privacy ones.

Video

Audio's cost structure plus image storage and bandwidth, multiplied by time. Rarely collected for its own sake in low-resource work.

Needs:

Recording and consent at scale, usually bundled with another goal (sign-language corpora, instructional content, or as source for extracted audio and frames).

Multimodal data

a combination

Not a separate modality but a pairing (text-image, audio-video, image-caption), collected so the modalities can be analyzed together.

Needs:

Both modalities for the same underlying item, often from different sources, with careful tracking of which pairs go together.

What a modality is

A **data modality** is the format a piece of data takes: text, image, audio, video, or some structured combination of these. Each modality encodes information differently and needs different tools to capture, store, and process it. A sentence and a photograph might describe the same painting, but one is a sequence of discrete symbols and the other is a dense grid of pixel values; nothing about how you collect, store, or validate one transfers directly to the other.

This matters more than it sounds. Most of the hard choices in dataset creation (where the data comes from, what it costs, who can label it, what can go wrong) are downstream of the modality decision, not the modeling decision. A multilingual sentiment project that decides to add a speech component is not "adding a feature." It is starting an entirely separate collection effort, often with its own budget line, its own legal questions, and its own pool of qualified contributors.

SCOPE OF THIS SECTION

This page covers modality at the **collection** stage: what each modality is and why it changes where you go to find data, who can supply it, and how much it costs. Designing annotation tasks and label schemas for a given modality (bounding boxes, transcription guidelines, emotion taxonomies) is covered later, in the modality sections: [Text](#), [Speech](#), [Vision](#) and [Multimodal](#).

The modalities you'll most often collect

Text

The most common starting point, and the one with the most existing tooling: web crawls, government archives, books, social media. For most African languages, though, text is also the modality where apparent abundance is most deceptive. [Joshi et al. \(2020\)](#) and [Kreutzer et al. \(2022\)](#), discussed in the [Introduction](#), found that a large share of what looks like African-language text online is mislabelled, machine-translated, or not the target language at all. Text "exists," in the sense that you can find files containing it; whether it exists in a form worth training on is a separate, harder question.

Speech and audio

Audio is often the *more* natural modality for many African languages, which carry centuries of oral tradition and, in some cases, no widely used standard orthography. Collecting it well usually means recording, not scraping: radio archives, community recording campaigns, phone-based data collection, or partnerships with broadcasters. It is more expensive per hour than text per word, both because recording requires equipment and consent, and because transcription afterward is itself a specialist annotation task.

Images

Image data splits into two different collection problems depending on whether you're sourcing existing images (museum archives, public photo collections, satellite imagery) or capturing new ones (camera rigs, mobile photo campaigns). The first raises licensing and redistribution questions; the second raises consent and privacy questions, especially where faces or identifiable locations are involved.

Video

Combines the cost structure of audio (recording, consent) with the storage and bandwidth cost of images, multiplied by time. Video is rarely collected for its own sake in low-resource-language work; it usually appears bundled with another goal: sign language corpora, instructional content, or as source material from which audio and frames are later extracted.

Multimodal data

Strictly speaking this isn't a separate modality but a *combination*: paired text-image, audio-video, or image-caption data, collected so the modalities can be analyzed together or against each other. The collection problem here compounds: you need both modalities to exist for the same underlying item, often from different sources, and you need to track which pairs go together. The [case study](#) later in this chapter walks through exactly this problem for a 28-language image-caption-emotion dataset.

Why this decision comes first

Before settling on sources or methods (the next two sections), it's worth answering, in writing, for your own project:

- **Which modality (or modalities) does the task actually require?** Don't default to text because it's familiar; a sentiment task on an oral-tradition language may be better served starting from audio.
- **Does the modality exist in usable form for your target language(s) at all**, or does "collection" really mean "creation": recording or writing things that don't yet exist anywhere?

- **Who can supply this modality?** Text can sometimes be sourced by people who don't speak the language well (a scraper doesn't need to read Hausa to download a Hausa web page); audio and video collection almost always requires native speakers in the room or on the call.
- **What does failure look like for this modality?** For text, it's silent: a mislabelled or machine-translated file looks identical to genuine data until someone checks. For audio, failure is often loud and visible (bad recording quality, wrong dialect) but harder to fix after the fact.

Get the modality decision right and the rest of this chapter (sources, scraping, APIs) becomes a menu of options. Get it wrong, and no amount of good tooling downstream will fix a collection plan built around the wrong kind of data.

 [Cite this page](#)  4 min read

CONTRIBUTORS  (<https://github.com/abumafri>)  (<https://github.com/shmuhammad>)  (<https://github.com/seyyaw>)

Data Sources

Learn the main categories of places raw data comes from, what each is good and bad at, and how to choose between them before committing to a collection method.

Why "source" and "method" are different questions

It's easy to conflate *where data comes from* with *how you get it*. They're separate decisions. "Twitter" is a source; "calling its API" is a method. "A national newspaper archive" is a source; "scraping its website" and "requesting a bulk export" are two different methods for reaching the same source. This page maps the sources; the next two pages, [Web Scraping](#) and [Application Programming Interfaces](#), go deep on the two methods used most often to reach them.

The main categories

The open web

News sites, blogs, forums, encyclopedias. Reachable by scraping or, where available, by API. For African languages this is the source most people reach for first and, as the [Introduction](#) shows, the one that disappoints most often, not because the web is poorly indexed, but because the underlying text in many target languages barely exists there in the first place.

Institutional and government archives

Court records, parliamentary proceedings, census documents, ministry publications. Often higher-quality and more consistently in the target language than open-web text, since institutions are required to publish in official or regional languages. Access is usually slower (requests, sometimes fees) but the resulting text tends to need far less cleaning.

Community and oral sources

Interviews, recorded proverbs, folktales, radio call-in shows, community meetings: sources that exist because people are talking, not because someone wrote something down. This is often the *only* viable source for languages without a strong written tradition, and it's where the playbook's core principle bites hardest: this data cannot be collected well by people outside the language community. Recruiting through tools like [AfriFinder](#) is built for exactly this category.

Broadcast and media archives

Radio and television archives, where accessible, sit between "scraped from the open web" and "recorded from scratch": the content already exists in spoken form, often in a standard register, and broadcasters sometimes hold transcripts or subtitles that double as parallel text. Partnership and licensing conversations with broadcasters take time, so this source rewards starting early.

Existing datasets and corpora

Don't discount work that's already been done. Datasets like [AfriSenti](#) for sentiment, or large multilingual collections built from community-translated seed data, exist precisely so the next project doesn't have to start from zero for a given language or task. Always check the license and intended use before reusing or redistributing.

Religious and literary texts

Bible translations, hymnals, and similar texts (distributed through projects like JW300 or BibleNLP) cover an unusually large number of low-resource languages, often more than any other parallel-text source. They're genuinely useful, especially as a bootstrap, but come with a well-documented register problem: religious text is formal, archaic, and topically narrow, and a model trained only on it will reproduce that register far outside where it belongs. Treat it as a starting point, not a representative sample.

Purpose-built collection campaigns

Sometimes nothing usable exists and the honest answer is: someone has to go create it (recording campaigns, structured interviews, mobile-phone data collection drives). This is the most expensive category per item and the one [Cost and Resource Planning](#) is built around, but it's also the only category where you control register, consent, and representativeness from the start.

Weighing sources against each other

For any candidate source, score it against the same handful of questions before committing time to it:

Question	Why it matters
What's the license, and can the result be redistributed?	Determines whether your eventual dataset can be released at all
Is consent already documented, or does it need to be obtained?	Shapes both ethics review and timeline
Does it cover the register and topics your task needs, or just one narrow slice (e.g. religious, formal, news)?	A source can be large and still badly unrepresentative
How much cleaning will the raw output need before it's usable?	Institutional archives are usually cleaner per word collected than open-web scrapes
Is this source stable, or could access change or disappear?	See Application Programming Interfaces for what happens when it does

No single source is "best" in the abstract; the right choice depends on the modality ([Data Modalities](#)), the language's resource tier, and what register the downstream task actually needs.

 [Cite this page](#)  4 min read

CONTRIBUTORS



(<https://github.com/abumafrim>)



(<https://github.com/shmuhammad>)



(<https://github.com/seyyaw>)

Web Scraping

Learn what web scraping is actually good for, where it falls short for African languages, and the legal, ethical, and practical considerations that should shape any scraping plan.

What scraping is, and what it isn't

Web scraping means programmatically extracting content from web pages, as opposed to using a platform's official API (the next section), or requesting a bulk export from an institution. It's attractive because it's cheap to start and doesn't require anyone's permission to begin (though, as below, that doesn't mean permission isn't needed). It's the most common entry point into data collection for exactly that reason.

It's also, by itself, the weakest tool in this chapter for the playbook's core problem. As the [Introduction](#) lays out, when [Kreutzer et al. \(2022\)](#) audited the large multilingual web crawls most NLP pipelines are built on, they found that for many low-resource languages a large share of the supposedly in-language text was mislabelled, machine-translated, or not the target language at all. Scraping more of the web doesn't fix that; it just collects more of the same noise faster. Scraping is a method for reaching a source that already has volume and quality. It cannot manufacture either where they don't exist.

The access landscape keeps shifting

Scraping plans that assume the open web behaves the way it did five years ago should be reconsidered. A large and growing share of sites now explicitly disallow AI-related crawlers in their `robots.txt` files, and several major platforms have moved to metered or paid access for bulk crawling rather than leaving it open. Legal disputes over scraping and AI training data remain active and unresolved in multiple jurisdictions ([Brown et al., 2024](#)). None of this makes scraping impossible, but it means:

- A scraping plan written today may not work the same way in a year. Budget time to re-check access periodically through a project, not just at the start.
- `robots.txt` and a site's terms of service aren't just etiquette; they're the basis on which a site operator, or a court, will judge whether your collection was legitimate. Read and respect them.
- A site that explicitly blocks crawling is signalling something about how it wants its content used. Treat that as a reason to look elsewhere rather than a technical obstacle to route around.

A framework for deciding whether and how to scrape

[Brown et al. \(2024\)](#) lay out a four-part framework for research scraping that maps directly onto the questions a contributor to this playbook should ask before starting:

- **Legal:** Does the site's terms of service prohibit automated access? Is the content copyrighted, and would your intended redistribution plan need a license or fall under an exception?
- **Ethical:** Did the people who created this content have any reasonable expectation about how it would be used? Scraping a government gazette is different from scraping a personal blog or a forum where people were writing for a small, known audience.

- **Institutional:** Does your institution's ethics review process (IRB or equivalent) need to sign off, especially if the content could identify a person?
- **Scientific:** Will the resulting sample be representative of what you actually need, or just of what was easiest to scrape? A crawl biased toward formal news sites will under-represent the everyday, conversational register most NLP tasks actually need.

Practical guidance

- **Respect `robots.txt` and rate limits.** Aggressive crawling can get your IP blocked, and for a site already wary of AI scraping, it confirms exactly the behavior they were trying to keep out.
- **Prefer official archives and bulk-export options over scraping live pages,** where they exist. They're more stable, less likely to break your pipeline mid-collection, and easier to defend later as a collection method.
- **Document everything as you go:** the exact URL pattern, the date of collection, the scraper version, and any filtering applied. This becomes the backbone of [Data Provenance and Traceability](#). Provenance is much harder to reconstruct after the fact than to log at the time.
- **Don't scrape behind logins or around anti-bot measures.** Circumventing access controls carries real legal risk and is a hard line many institutional review boards will not approve, regardless of the research value.
- **For African-language text specifically, validate language identification on anything you scrape.** Off-the-shelf language identifiers are themselves trained mostly on high-resource languages and frequently misclassify African-language text, exactly the failure mode Kreutzer et al. documented at scale. Spot-check a sample with a native speaker before trusting an automated filter.

A minimal, well-behaved scraper

The points above translate into code more directly than they might seem to. A scraper that respects `robots.txt`, rate-limits itself, records provenance, and verifies the language of what it collects is not much longer than a careless one. The example below uses only widely available libraries (`requests`, `beautifulsoup4`, and a language identifier).

```

import time
import json
import urllib.robotparser
from datetime import datetime, timezone
from urllib.parse import urlparse

import requests
from bs4 import BeautifulSoup

USER_AGENT = "AfriPlaybook-collector/1.0 (+contact: you@example.org)"
RATE_LIMIT_SECONDS = 2.0 # one request every two seconds, at most

def allowed_by_robots(url: str) → bool:
    """Honour robots.txt before fetching. If we can't read it, don't fetch."""
    parts = urlparse(url)
    robots_url = f"{parts.scheme}://{parts.netloc}/robots.txt"
    parser = urllib.robotparser.RobotFileParser()
    try:
        parser.set_url(robots_url)
        parser.read()
    except Exception:
        return False
    return parser.can_fetch(USER_AGENT, url)

def fetch(url: str) → str | None:
    if not allowed_by_robots(url):
        print(f"Skipped (robots.txt disallows): {url}")
        return None
    response = requests.get(url, headers={"User-Agent": USER_AGENT}, timeout=30)
    response.raise_for_status()
    time.sleep(RATE_LIMIT_SECONDS) # be a good guest
    return response.text

def extract_text(html: str) → str:
    soup = BeautifulSoup(html, "html.parser")
    for tag in soup(["script", "style", "nav", "footer"]):
        tag.decompose()
    return " ".join(soup.get_text(separator=" ").split())

def collect(urls: list[str], expected_lang: str, out_path: str) → None:
    """Fetch each URL, keep only text that looks like the target language,
    and record provenance for every record we keep."""
    from glotlid import GlotLID # pip install glotlid

    identifier = GlotLID()
    with open(out_path, "w", encoding="utf-8") as out:
        for url in urls:
            html = fetch(url)
            if html is None:
                continue
            text = extract_text(html)
            label, score = identifier.predict(text) # e.g. ("hau_Latn", 0.97)
            if not label.startswith(expected_lang) or score < 0.7:
                print(f"Skipped (language {label}, score {score:.2f}): {url}")
                continue
            record = {
                "text": text,
                "source_url": url,
                "language": label,
                "lang_confidence": round(float(score), 3),
                "collected_at": datetime.now(timezone.utc).isoformat(),
                "collector": USER_AGENT,
            }
            out.write(json.dumps(record, ensure_ascii=False) + "\n")

if __name__ == "__main__":
    collect(
        urls=["https://example.org/hausa-article-1"],
        expected_lang="hau", # ISO 639-3 prefix for Hausa
    )

```

```
    out_path="raw_hausa.jsonl",  
)
```

Three things in this script do the work the prose asked for. The `allowed_by_robots` check refuses to fetch anything the site disallows, and fails closed if the policy can't be read. The `RATE_LIMIT_SECONDS` pause keeps the crawl from looking like an attack. And the language check, using [GlotLID](#) here, drops anything that isn't confidently the target language, which is the single most important filter for African-language scraping given how often general-purpose crawls mislabel it ([Kreutzer et al., 2022](#)). The confidence threshold of `0.7` is a starting point, not a universal value: set it by spot-checking a sample with a native speaker, as described above.

When scraping is the right call

Scraping earns its place when the target source is large, public, has a clear and permissive access policy, and, critically, actually contains meaningful volume in the target language. It's the wrong tool when the honest problem is that the language simply isn't well represented online yet. In that case, the community, institutional, and purpose-built collection routes covered in [Data Sources](#) are where the real work is.

 [Cite this page](#) ⌚ 6 min read

CONTRIBUTORS  (<https://github.com/abumafrim>)  (<https://github.com/shmuhammadd>)

Application Programming Interfaces (APIs)

Learn how API-based collection differs from scraping, what it's well suited for, and why API access is one of the least stable foundations a data collection plan can be built on.

How this differs from scraping

An API (Application Programming Interface) is a structured, sanctioned channel a platform provides for accessing its data, as opposed to extracting it from rendered web pages. Where scraping takes what's publicly visible and infers structure from a page's HTML, an API hands you already-structured data (a tweet object, a Wikipedia article's wikitext, a government dataset's rows) directly, usually with documentation, authentication, and explicit rate limits.

This makes APIs more reliable to build a pipeline around in the short term: the data is cleaner, the schema is documented, and you're not fighting page-layout changes. It does not make them more reliable in the long term, for reasons worth taking seriously before you commit a project's collection plan to one.

A cautionary, very real example

AfriSenti, the sentiment dataset behind SemEval-2023's first Afrocentric shared task, was built on more than 110,000 tweets across 14 African languages, gathered through Twitter's then-free API access for academic research ([Muhammad et al., 2023](#)). That access model no longer exists. In 2023, Twitter (now X) ended free academic API access and introduced paid tiers, with enterprise-level access priced at tens of thousands of dollars per month, far beyond what most academic projects can pay ([Brown et al., 2024](#)). A project planned today that assumed it could reproduce AfriSenti's collection process, on the same terms, simply could not.

This isn't a story about one platform behaving badly. It's the generic risk of any API: it's a privilege the platform grants, not a right you hold, and it can be repriced, restricted, or revoked with little notice, for reasons that have nothing to do with you or your research. Plan accordingly.

What APIs are well suited for

- **Open and public-data APIs**, where governments or international bodies expose structured datasets (census, agricultural, health, climate data) under an explicit open license. These tend to be the most stable category, since the publishing motivation (transparency, compliance) doesn't depend on platform monetization.
- **Wikimedia APIs** (Wikipedia, Wiktionary, Wikidata), which cover a surprising number of African languages, even some with otherwise very little written presence online, and are explicitly built for reuse.
- **National broadcaster and library APIs**, where they exist, often as the structured counterpart to the broadcast archives discussed in [Data Sources](#).
- **Social media APIs**, with eyes open: useful for register and topical diversity that's hard to get elsewhere, but the category most exposed to the pricing and access risk described above.

Practical guidance

- **Read the terms of service before you read the API documentation.** What you're allowed to do with the data you collect (store it, redistribute it, publish it as part of a dataset) is governed by the ToS, not by what the API technically lets you request.
- **Archive what you collect immediately**, in its raw form, rather than re-fetching from the API later. If access changes or disappears, your archived pull is what you have; assume you may not be able to go back for more.
- **Log the exact endpoint, parameters, API version, and date of every pull.** This is the same provenance discipline as scraping (see [Data Provenance and Traceability](#)), and it matters more here, because an API's behavior can change silently between calls in a way a static web page can't.
- **Respect rate limits and use backoff.** Getting your access revoked for abuse, even unintentionally, can end a collection effort outright, unlike a scraper simply being slowed down.
- **Don't design a project's entire data plan around a single platform's API.** Treat any one API as one source among several, and have a sense of what the project does if that source disappears mid-project. As the AfriSenti example shows, it has happened before, with no advance warning.

A worked example: pulling from a Wikimedia API

Of the categories above, the Wikimedia APIs are the safest to learn on: they are built for reuse, cover many African languages, and are not going to reprice overnight. The example below pulls article extracts from a language edition of Wikipedia, archives the raw response exactly as received, and records provenance for each article. Swap `lang` for any Wikipedia language code, such as `yo` (Yoruba), `sw` (Swahili), `ha` (Hausa), or `am` (Amharic).

```

import json
from datetime import datetime, timezone

import requests

USER_AGENT = "AfriPlaybook-collector/1.0 (+contact: you@example.org)"

def pull_extracts(lang: str, titles: list[str]) → list[dict]:
    """Fetch plain-text extracts for given article titles from one
    Wikipedia language edition, archiving raw responses and provenance."""
    endpoint = f"https://{lang}.wikipedia.org/w/api.php"
    params = {
        "action": "query",
        "format": "json",
        "prop": "extracts",
        "explaintext": "1",          # plain text, not HTML
        "titles": "|".join(titles),
        "formatversion": "2",
    }
    response = requests.get(
        endpoint, params=params, headers={"User-Agent": USER_AGENT}, timeout=30
    )
    response.raise_for_status()
    pulled_at = datetime.now(timezone.utc).isoformat()

    # Archive the raw response first, before any processing.
    with open(f"raw_{lang}_response.json", "w", encoding="utf-8") as raw:
        json.dump(response.json(), raw, ensure_ascii=False, indent=2)

    records = []
    for page in response.json()["query"]["pages"]:
        if "extract" not in page:      # missing or redirected pages
            continue
        records.append({
            "title": page["title"],
            "text": page["extract"],
            "language": lang,
            "source_url": f"https://{lang}.wikipedia.org/wiki/{page['title'].replace(' ', '_')}",
            "license": "CC BY-SA 4.0", # Wikipedia text license
            "api_endpoint": endpoint,
            "pulled_at": pulled_at,
        })
    return records

if __name__ == "__main__":
    records = pull_extracts(lang="yo", titles=["Nàìjíríà", "Èkó"])
    with open("wikipedia_yo.jsonl", "w", encoding="utf-8") as out:
        for record in records:
            out.write(json.dumps(record, ensure_ascii=False) + "\n")

```

The order of operations matters here. The raw response is written to disk before anything is extracted from it, so that if the schema turns out to differ from what was expected, or access changes later, the original pull is preserved and the parsing can be redone offline. Each record carries its license ([CC BY-SA 4.0](#) for Wikipedia text, which means downstream reuse must attribute and share alike), the exact endpoint, and the pull time. That is the provenance discipline from [Data Provenance and Traceability](#) applied at the point of collection, where it is cheap, rather than reconstructed afterwards, where it is often impossible.

The bigger pattern

Both this page and [Web Scraping](#) point at the same lesson from different directions: methods that depend on a third party's continued goodwill (whether that's a platform's API pricing or a site's tolerance for crawlers) are inherently less stable than methods that don't. Community-sourced data, institutional partnerships, and purpose-built collection

campaigns (see [Data Sources](#)) take longer to set up, but they don't disappear because someone else changed a pricing page.

 [Cite this page](#)  5 min read

CONTRIBUTORS  (<https://github.com/abumafrim>)  (<https://github.com/shmuhammadd>)

Case Study: Multimodal Data Collection

A worked example of what happens when a collection plan has to satisfy two modalities (and dozens of languages) at the same time, using ArtELingo-28 ([Mohamed et al., EMNLP 2024 ↗](#)) as a concrete reference point.

The task

ArtELingo-28 is a benchmark built to study how people from different cultures respond emotionally to the same piece of visual art, across 28 languages. For every image, annotators in many languages independently chose the emotion the artwork evoked and wrote a short caption explaining why. The result is roughly 200,000 annotations over 2,000 WikiArt images (around 140 annotations per image, distributed across the 28 languages), making it possible to compare, image by image, whether annotators in two different languages read the same painting the same way.

This is a genuinely multimodal collection problem: the image layer and the text layer come from completely different places, are governed by different rules, and have to be collected in a way that keeps them aligned.

Decision 1: the image layer was reused, not collected fresh

The images themselves weren't newly photographed or scraped for this project; they're drawn from WikiArt, an existing public art archive, the same source used by predecessor projects in this line of work. This illustrates a point from [Data Modalities](#): not every modality in a multimodal project needs to be collected the same way, or even collected at all. Here, the image modality was sourced from an existing, curated archive; the text/annotation modality was the part that genuinely needed new collection. Recognizing which layer is "already solved" and which layer is the actual collection problem saves significant time and budget.

Reusing existing images instead of collecting new ones doesn't remove the licensing question; it relocates it. Any project building on a third-party image archive has to work out, separately from its annotation collection plan, what redistribution rights it has for the images once they're bundled into a newly released dataset alongside new captions and labels. That question sits squarely in the territory covered later in [Documentation, Data Release, and Governance ↗](#), but it has to be answered during the collection-planning stage, not discovered after annotation is finished.

Decision 2: cross-cultural comparison sets a constraint before annotation starts

The entire scientific premise, that you can compare how different language communities respond to "the same" artwork, depends on every language annotating the same underlying image set. That sounds obvious, but it has to be designed into the collection plan from day one. If a future, larger version of this kind of project draws its images from sources with looser licensing rather than a fixed, pre-cleared set, it has to confirm that image availability is consistent across every language's split *before* annotation begins, not after. A test set that's accidentally missing a handful of images in one language's annotation batch quietly breaks the comparison the whole project is built to support. This is a collection-stage decision masquerading as an annotation-stage problem. By the time annotators are working, it's too late to fix cheaply.

Decision 3: cost and labor planning had to scale by language, not just by image

For a single-language captioning dataset, the unit of collection cost is roughly "per image." Here, the unit is "per image, per language": 2,000 images annotated independently in each of 28 languages is a fundamentally different scale of labor than 2,000 images annotated once. Annotators were recruited through Amazon Mechanical Turk and compensated 0.10 *per annotation* (roughly 8/hour, given an average of about 45 seconds per task), and the project secured institutional review board approval and informed consent ahead of annotation, given its use of paid crowdworkers. None of those numbers would have been knowable without first deciding the multiplier (28 languages) at the collection-planning stage, which is exactly the work described in [Cost and Resource Planning](#).

Takeaways for your own multimodal project

- **Inventory each modality separately before planning collection.** Decide, layer by layer, which modality already exists in usable form somewhere (reuse it) and which has to be newly created (budget for it properly).
- **If cross-X comparison is the scientific point of the project** (cross-cultural, cross-language, cross-time), **the comparability constraint is a collection-stage design decision**, not something to verify after the fact. Lock down what stays constant across every split before recruiting a single annotator.
- **Reusing existing assets, like an image archive, doesn't remove licensing work; it moves the question** from "can I collect this" to "can I redistribute this bundled with what I add on top."
- **When a project multiplies across many languages, the cost driver isn't the dataset size; it's the multiplier.** Plan budget and timeline around that multiplier explicitly, rather than as a vague "and then do it for the other languages too" afterthought.



CONTRIBUTE YOUR OWN CASE STUDY

This page covers one published example. If you've run a multimodal or cross-lingual collection effort, successful or not, [write it up](#) as an additional case study. What went wrong is often more useful to the next person than what went right.

 [Cite this page](#)  4 min read

CONTRIBUTORS



(<https://github.com/abumafrim>)



(<https://github.com/shmuhammadd>)



(<https://github.com/seyyaw>)

Cost and Resource Planning

Learn how to effectively plan the resources required for dataset creation, including budgeting, timelines, and scaling strategies.

Why Planning Matters

Dataset creation can be resource-intensive. Proper planning helps ensure efficient use of time, budget, and human effort while maintaining data quality.

Key Components of Planning

Budgeting Annotation Costs

- **Annotation cost estimation** – Calculate cost per sample or per task
- **Workforce planning** – Consider expert vs crowd annotators
- **Tooling costs** – Include platforms, storage, and infrastructure
- **Quality control costs** – Account for validation and review processes

Time Estimation

- **Task complexity** – More complex tasks require more time per annotation
- **Annotator speed** – Estimate based on pilot studies or benchmarks
- **Project phases** – Include setup, training, annotation, and validation
- **Buffer time** – Plan for delays and iterations

Scaling Strategy

- **Incremental scaling** – Start small and expand gradually
- **Automation support** – Use tools to speed up preprocessing and validation
- **Parallel workflows** – Distribute tasks across multiple annotators
- **Quality vs scale balance** – Maintain data quality while increasing size

 [Cite this page](#)  1 min read

CONTRIBUTOR



(<https://github.com/abumafrim>)

Data Cleaning and Preprocessing

Learn how to prepare raw data for use in language AI systems by improving quality, consistency, and usability.

Why Data Cleaning Matters

Raw data often contains noise, inconsistencies, and errors. Proper cleaning and preprocessing ensure that datasets are reliable, accurate, and suitable for downstream tasks such as training and evaluation.

Key Steps in Data Cleaning

Deduplication, Normalization, and Filtering

- **Deduplication** – Remove duplicate entries to avoid bias and overrepresentation
- **Normalization** – Standardize text (e.g., casing, punctuation, encoding)
- **Filtering** – Remove irrelevant, low-quality, or out-of-scope data

Language Detection and Formatting

- **Language detection** – Identify and verify the language of each data instance
- **Formatting** – Ensure consistent structure (e.g., JSON, CSV, text fields)
- **Encoding consistency** – Maintain uniform character encoding (e.g., UTF-8)

Noise and Toxicity Handling

- **Noise removal** – Clean unwanted artifacts such as HTML tags, emojis (if not needed), or corrupted text
- **Toxicity handling** – Detect and manage harmful, offensive, or unsafe content depending on project goals

Missing and Corrupted Data Handling

- **Missing data** – Identify incomplete entries and decide whether to fill, ignore, or remove them
- **Corrupted data** – Detect broken or unreadable content and clean or discard it
- **Validation checks** – Ensure data integrity after preprocessing

A cleaning pipeline, end to end

The steps above are easier to keep honest when they are written as one pass over the raw records. The function below takes the JSONL produced by the collectors in [Web Scraping](#) and [APIs](#), then normalizes, deduplicates, language-filters, and length-filters in a single pass, writing both the cleaned data and a small report of what was dropped and why.

```

import json
import re
import unicodedata
from collections import Counter

def normalize(text: str) → str:
    """NFC-normalize Unicode, collapse whitespace, strip control characters.
    NFC matters for African scripts: the same Yoruba or Amharic character can
    be encoded as one codepoint or as a base plus combining marks, and the two
    forms will not deduplicate or match unless normalized first."""
    text = unicodedata.normalize("NFC", text)
    text = "".join(ch for ch in text if unicodedata.category(ch)[0] ≠ "C")
    return " ".join(text.split())

def clean(in_path: str, out_path: str, expected_lang: str,
         min_words: int = 5) → None:
    from glotlid import GlotLID

    identifier = GlotLID()
    seen: set[str] = set()
    dropped = Counter()
    kept = 0

    with open(in_path, encoding="utf-8") as src, \
         open(out_path, "w", encoding="utf-8") as out:
        for line in src:
            record = json.loads(line)
            text = normalize(record.get("text", ""))

            if len(text.split()) < min_words:
                dropped["too_short"] += 1
                continue

            fingerprint = text.casefold()
            if fingerprint in seen:          # exact-duplicate removal
                dropped["duplicate"] += 1
                continue
            seen.add(fingerprint)

            label, score = identifier.predict(text)
            if not label.startswith(expected_lang) or score < 0.7:
                dropped[f"wrong_language:{label}"] += 1
                continue

            record["text"] = text
            record["language"] = label
            out.write(json.dumps(record, ensure_ascii=False) + "\n")
            kept += 1

    print(f"Kept {kept} records.")
    for reason, count in dropped.most_common():
        print(f"Dropped {count}: {reason}")

if __name__ == "__main__":
    clean("raw_hausa.jsonl", "clean_hausa.jsonl", expected_lang="hau")

```

The duplicate check here is exact-match, which is enough for most collection runs. When near-duplicates are a concern, for example the same news wire republished across several sites with minor edits, reach for MinHash with a library such as [datasketch](#) rather than comparing every pair, which does not scale. The printed report is not an afterthought: a cleaning step that silently drops most of its input is usually a sign of a wrong language code or a bad threshold, and you only notice if the counts are in front of you.

A note on the cleaned format

Keeping cleaned data as JSONL, one record per line, is the most practical default for African-language work. It streams without loading the whole file into memory, it survives being appended to, and every record carries its own provenance fields rather than relying on a separate manifest. A single cleaned record looks like this:

```
{
  "text": "Kano ita ce cibiyar kasuwanci a arewacin Najeriya.",
  "language": "hau_Latn",
  "source_url": "https://example.org/hausa-article-1",
  "license": "CC BY-SA 4.0",
  "collected_at": "2026-02-14T09:31:00+00:00"
}
```

CSV is a reasonable alternative for purely tabular data with no embedded newlines, but it handles multi-line text and commas-in-text poorly, and the diacritics common in African orthographies are more reliably preserved in UTF-8 JSON than in spreadsheets that may re-encode on save. Whatever the format, write it as UTF-8 and never let a tool silently downgrade the encoding.

 [Cite this page](#)  4 min read

CONTRIBUTOR



(<https://github.com/abumafrim>)

Data Provenance and Traceability

Learn how to track the origin, history, and transformations of your data to ensure transparency, reproducibility, and accountability.

Why Provenance Matters

Understanding where data comes from and how it has been processed is essential for building trustworthy datasets. Provenance supports reproducibility, enables auditing, and helps identify potential issues in data quality and bias.

Key Components of Provenance

Source Tracking

- **URLs and references** – Record links or original sources of the data
- **Contributors** – Track who collected, created, or provided the data
- **Collection context** – Document when, where, and how the data was obtained

Data Lineage

- **Data evolution** – Track how data changes over time
- **Versioning** – Maintain different versions of datasets
- **Pipeline tracking** – Document each stage of data processing

Transformation Logs

- **Preprocessing steps** – Record cleaning, normalization, and filtering operations
- **Annotation processes** – Track labeling methods and guidelines used
- **Modifications** – Log any changes made to the data after collection
- **Audit trails** – Maintain records for reproducibility and verification

A dataset card you can version

Per-record provenance, like the fields the collectors and cleaner attach to each line, answers "where did this row come from". A dataset card answers the same question for the dataset as a whole, in a single file that lives in the repository next to the data and changes with it. Writing it as YAML keeps it readable, diff-friendly in version control, and parseable by tooling. This format follows the spirit of the documentation standards covered in [Documentation](#), kept short enough to actually maintain.

```
# dataset-card.yaml
name: hausa-news-2026
version: 1.1.0
languages:
  - hau_Latn      # Hausa, Latin script (ISO 639-3 + script)
modality: text
task: text-classification
size:
  records: 18432
  collected_records: 24901  # before cleaning, for an honest drop rate

sources:
  - name: Example Hausa news portal
    url: https://example.org
    method: web-scraping
    license: CC BY-SA 4.0
    access_date: 2026-02-14
  - name: Hausa Wikipedia
    url: https://ha.wikipedia.org
    method: wikimedia-api
    license: CC BY-SA 4.0
    access_date: 2026-02-15

processing:
  - step: normalize
    detail: Unicode NFC, whitespace collapse, control-char strip
  - step: deduplicate
    detail: exact case-folded match
  - step: language-filter
    detail: GlotLID, threshold 0.7, kept hau_Latn only

governance:
  consent: not-applicable      # public, already-published text
  owner: Example Community NLP Group
  contact: data@example.org
  pii_reviewed: true

citation: >
  Example Community NLP Group (2026). hausa-news-2026 (v1.1.0).
```

Two fields here earn their place specifically for African-language work. Recording `collected_records` alongside the final `size` makes the drop rate visible: if a cleaning pass kept only a fraction of what was collected, that belongs in the open, not hidden. And pinning the language as `hau_Latn`, with both the ISO 639-3 code and the script, avoids the ambiguity that bare codes like `ha` invite, which matters for languages written in more than one script.

A transformation log that travels with the data

The dataset card records the current state. A transformation log records the history, one appended line each time the data changes, so that a released version can always be traced back through every step that produced it. Appending to a JSONL log is enough, and it costs nothing to keep:

```
{
  "version": "1.0.0",
  "date": "2026-02-16",
  "action": "initial-collection",
  "records": 24901,
  "by": "A. Bello",
  "notes": "raw scrape + wikipedia pull"
}
{
  "version": "1.0.1",
  "date": "2026-02-18",
  "action": "cleaning",
  "records": 19120,
  "by": "pipeline v2",
  "notes": "normalize, dedup, language-filter"
}
{
  "version": "1.1.0",
  "date": "2026-03-02",
  "action": "manual-review",
  "records": 18432,
  "by": "native-speaker reviewers",
  "notes": "removed 688 mislabelled and off-topic records"
}
```

Read top to bottom, this log tells the whole story of the dataset: how much was collected, how much survived cleaning, and what human review changed. That is exactly the audit trail that makes a dataset defensible later, and it is far cheaper to append a line at each step than to reconstruct the history from memory at release time.

 [Cite this page](#)  3 min read

CONTRIBUTOR  (<https://github.com/abumafrim>)

Ethics, Bias, and Governance

Learn how to ensure responsible dataset creation by addressing bias, protecting privacy, and maintaining transparency throughout the data lifecycle.

Why Ethics and Governance Matter

Datasets directly influence the behavior of language AI systems. Poor handling of bias, privacy, or sensitive content can lead to harmful outcomes. Ethical practices and governance frameworks help ensure fairness, accountability, and trust.

Key Components of Ethical Data Practices

Bias Identification and Mitigation

- **Bias detection** – Identify imbalances or skewed representations in data
- **Source bias** – Assess biases introduced by data sources
- **Annotation bias** – Monitor inconsistencies across annotators
- **Mitigation strategies** – Apply re-sampling, re-weighting, or guideline refinement

PII Detection and Removal

- **Personal data identification** – Detect names, addresses, contact details, and identifiers
- **Automated detection tools** – Use models or rules to flag sensitive information
- **Manual review** – Validate automated detection with human checks
- **Data removal or masking** – حذف or obfuscate personal identifiers

Anonymization Strategies

- **De-identification** – Remove or replace identifiable information
- **Pseudonymization** – Substitute identifiers with artificial labels
- **Aggregation** – Present data in grouped form to prevent re-identification
- **Risk assessment** – Evaluate re-identification risks after anonymization

Sensitive Attribute and Content Handling

- **Sensitive attributes** – Gender, ethnicity, religion, health, or political views
- **Content moderation** – Handle harmful, offensive, or explicit content carefully
- **Access control** – Restrict sensitive data to authorized users
- **Use-case alignment** – Decide inclusion based on task requirements

Fair Representation

- **Inclusive sampling** – Ensure diverse representation across groups
- **Balanced datasets** – Avoid over- or under-representation

- **Context awareness** – Consider cultural and linguistic diversity
- **Evaluation fairness** – Test models across different subgroups

Risk Documentation and Transparency

- **Risk identification** – Document potential harms and limitations
- **Datasheets and documentation** – Provide clear dataset descriptions
- **Transparency practices** – Share collection, processing, and annotation details
- **Governance policies** – Define rules for dataset usage and distribution

 [Cite this page](#)  2 min read

CONTRIBUTOR  (<https://github.com/abumafrim>)

Annotation Task Design and Human Factors

The shape of an annotation task decides its quality long before the first label is applied. A task that is clear, well-scoped, and comfortable to do produces consistent data. One that overloads or exhausts the people doing it produces noise, no matter how good the guidelines are. Design the task for the human who will sit with it for hours.

Estimate task complexity before you scale

Every annotation decision carries a cognitive cost, and that cost compounds over thousands of items. A binary judgment, such as whether a text is offensive or not, is far cheaper than a multi-label one, such as assigning all applicable emotions from a set of eight, which is cheaper again than span-level work like marking every named entity and its type. Estimate the complexity honestly, because it drives both the budget and the error rate. Where a task is genuinely complex, break it into a sequence of simpler decisions rather than asking for everything at once. The Masakhane named-entity effort kept its per-language tasks tractable by giving each language its own small team working to one shared, simple protocol rather than a single sprawling labelling scheme ([Adelani et al., 2022](#)). Use the pilot to measure the real time per item, and let that number, not optimism, set the schedule.

Design against annotator fatigue

Quality falls as attention does. Long unbroken sessions, monotonous batches, and ambiguous items all wear annotators down, and tired annotators reach for the easy label rather than the right one. Plan for this with short batches, regular breaks, and rotation across task types where the project allows it. Fatigue is not only a quality problem. For tasks that expose annotators to hate speech, abuse, or other distressing material, which is common in African sentiment and safety datasets, prolonged exposure becomes a wellbeing problem, and it calls for content warnings, the freedom to skip or stop, workload limits, and access to support (covered in the annotation and [data governance](#) chapters). Treating annotation as skilled, sustainable work rather than disposable clicks is also what keeps a good team available for the next project ([Sambasivan et al., 2021](#)).

Let the interface do some of the work

The annotation tool is part of the task design, not a neutral container. A good interface removes whole classes of error before they happen. It constrains inputs to the allowed label set so a typo cannot become a new category, it shows the surrounding context an annotator needs to judge a word inside a sentence, and it uses keyboard shortcuts so the work is fast and the physical strain is low. Two constraints matter especially for African projects. Many annotators work on modest hardware over intermittent, metered connections, so the tool should be light, tolerant of dropped connections, and usable on a small screen. It should also handle the orthography of the target language properly, including diacritics and non-Latin scripts, so that what the annotator types is what gets stored. A task that is awkward to do will be done awkwardly.

A labeling configuration

These principles are not abstract: they are choices you make in the labeling configuration. The examples throughout this playbook use the Label Studio configuration format, an open XML format that several annotation tools support (including the playbook's companion tool, AfriAnnotate), so the configs here are portable rather than tied to any one

product. A config is a small, readable document that defines exactly what the annotator sees and what they are allowed to do. The example below sets up a single-label sentiment task of the kind behind AfriSenti (Muhammad et al., 2023).

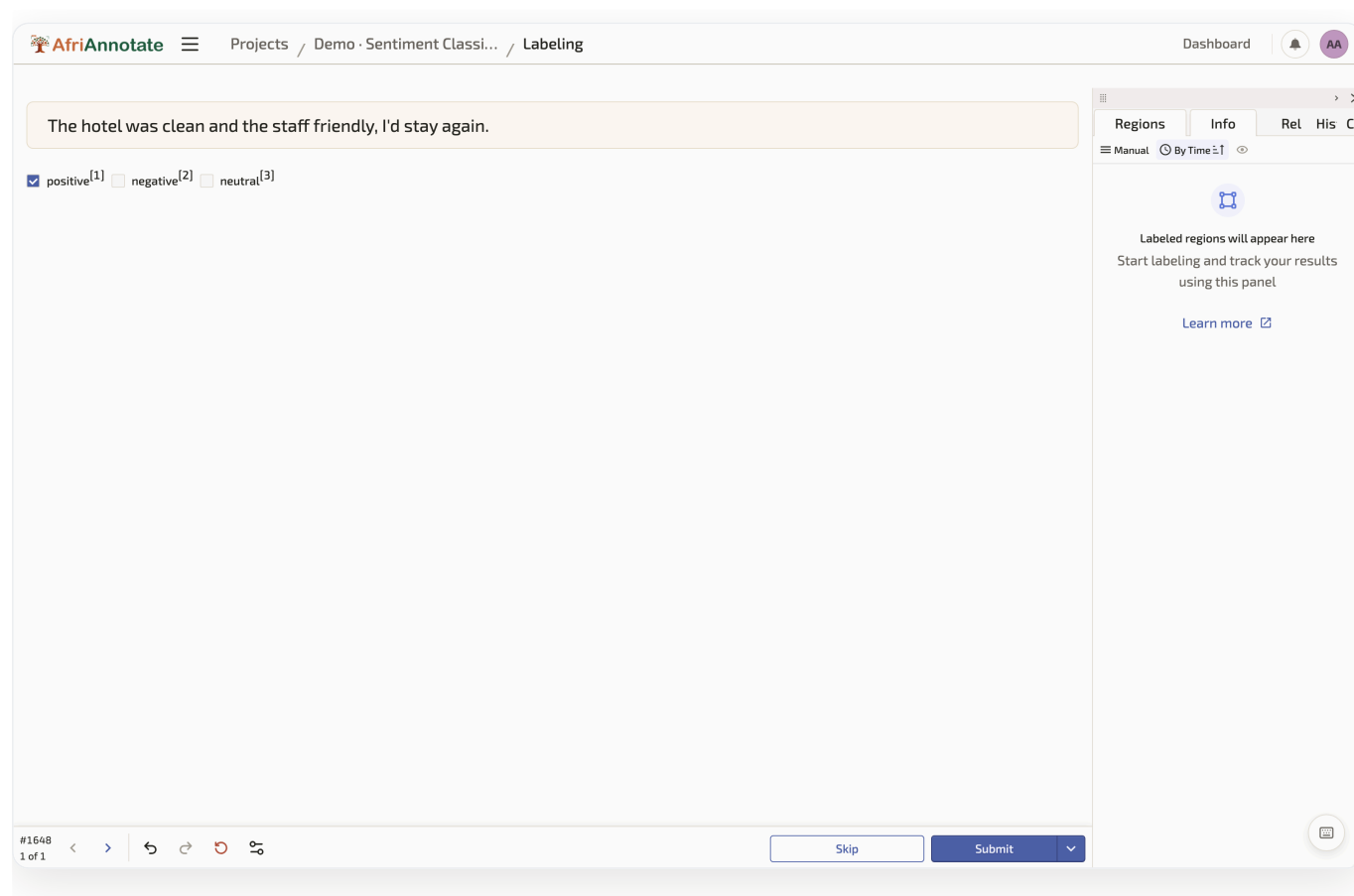
```
<View>
  ←!— The source text sits in its own card so the annotator can always tell
        the content apart from the instruction prompts and labels below it. →
  <View style="background:#FBF7F0; border:1px solid #E7DDCB; border-radius:8px; padding:14px 16px; margin-bottom:18px;">
    <Text name="text" value="$text"/>
  </View>

  ←!— One choice only, drawn from a fixed set: a typo cannot become
        a new category, and the decision stays a cheap binary-style call. →
  <Choices name="sentiment" toName="text" choice="single" required="true">
    <Choice value="Positive" hotkey="1"/>
    <Choice value="Neutral" hotkey="2"/>
    <Choice value="Negative" hotkey="3"/>
  </Choices>

  ←!— An explicit escape hatch, so a tired or unsure annotator skips
        rather than guessing. These get routed to adjudication. →
  <Choices name="flag" toName="text" choice="single">
    <Choice value="Code-mixed / wrong language" hotkey="8"/>
    <Choice value="Unclear, needs a second opinion" hotkey="9"/>
  </Choices>
</View>
```

Three of the design points above are enforced directly in this config. `choice="single"` with a fixed `<Choice>` set constrains the input to the allowed labels, so the interface, not the guideline document, prevents stray categories. The `hotkey` attributes let an annotator work entirely from the keyboard, which is both faster and gentler on the hands over a long session. And the `flag` block gives the explicit skip route the fatigue discussion calls for, rather than forcing a guess. Wrapping the text in its own styled `<View>` is a small touch that matters over thousands of items: it sets the content apart from the prompts and labels, so an annotator never confuses what they are reading with what they are being asked.

That configuration renders, for the annotator, as a clean single-task screen where the source text stands out from the question:



For a span-level task such as named-entity recognition, the same format expresses a harder job by swapping `<Choices>` for `<Labels>`, which lets the annotator select a stretch of text and tag it:

```
<View>
  <Labels name="entities" toName="text">
    <Label value="PER" hotkey="1" background="#e8743b"/>
    <Label value="ORG" hotkey="2" background="#19a979"/>
    <Label value="LOC" hotkey="3" background="#945ecf"/>
    <Label value="DATE" hotkey="4" background="#13a4b4"/>
  </Labels>
  <Text name="text" value="$text" />
</View>
```

The `$text` field in both configs maps straight onto the `text` field in the cleaned JSONL records from the [Data Collection](#) chapter, so the import step is just uploading that file: each line becomes one task, and the provenance fields ride along unchanged. Starting from a tiny config like this, and growing it only when a pilot shows it is needed, keeps the task tractable in the way the complexity discussion above recommends.

Selecting a span and tagging it as an entity, in the AfriAnnotate editor:

AfriAnnotate

Demo · Named-Entity Rec...

Project sections

Default

Actions

Columns

Filters

Order by

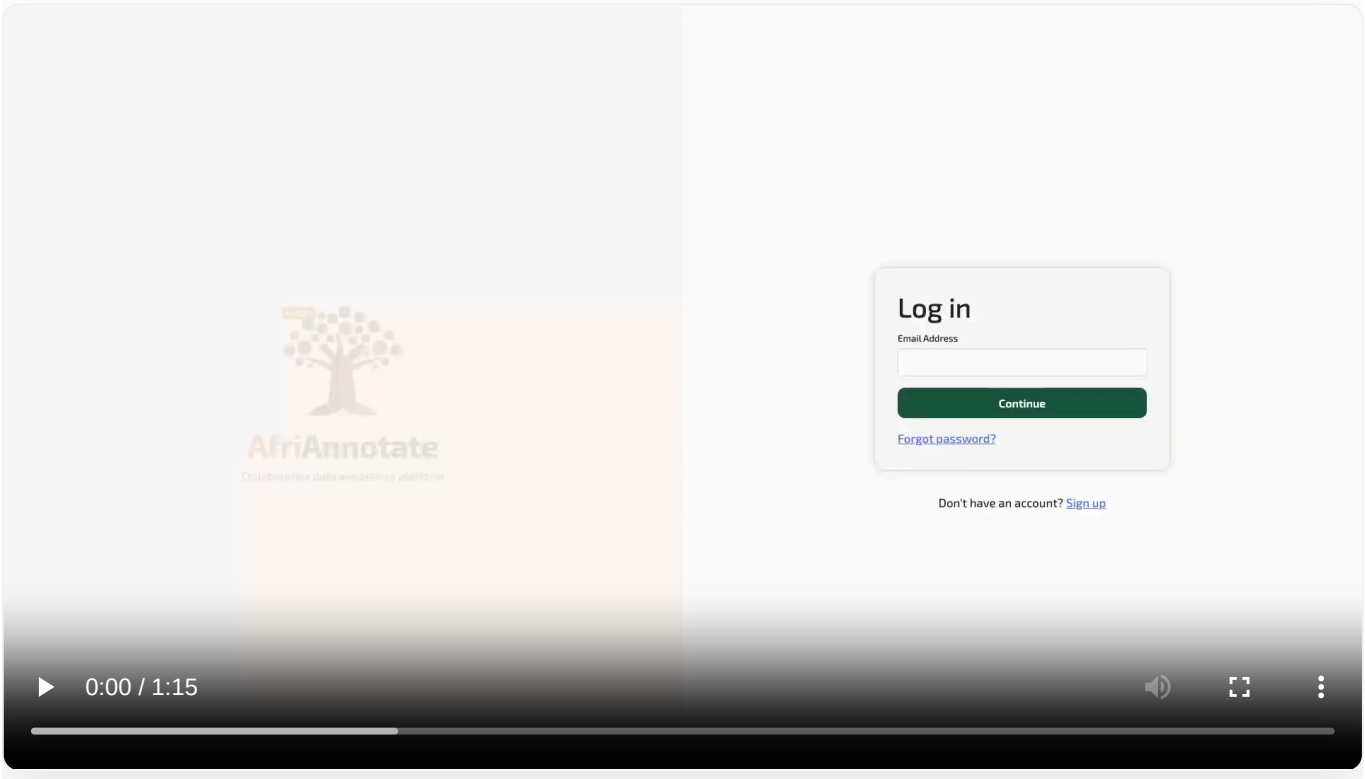
Label All Tasks

☐	ID	Completed	text	gold_entities	
☐	1810		Mount Kilimanjaro is the highest peak in Tanzania.	[{"text": "Mount Kilimanjaro", "label": "LOC"}, {"text": "Tanzania", "label": "L	{}
☐	1811		Nelson Mandela became president of South Africa in 1994.	[{"text": "Nelson Mandela", "label": "PER"}, {"text": "South	{}
☐	1812		The University of Lagos hosted the conference last June.	[{"text": "University of Lagos", "label": "ORG"}, {"text": "Last	{}
☐	1813		Kofi Annan led the United Nations for ten years.	[{"text": "Kofi Annan", "label": "PER"}, {"text": "United	{}
☐	1814		Victoria Falls sits on the border of Zambia and Zimbabwe.	[{"text": "Victoria Falls", "label": "LOC"}, {"text": "Zambia", "label": "LOC	{}
☐	1815		Burna Boy performed at Madison Square Garden last Friday.	[{"text": "Burna Boy", "label": "PER"}, {"text": "Madison Square	{}
☐	1816		Ethiopian Airlines will open a new hub in Lomé next year.	[{"text": "Ethiopian Airlines", "label": "ORG"}, {"text": "Lomé", "label": "LOC"},	{}
☐	1817		Wole Soyinka is a Nigerian playwright based in Abeokuta.	[{"text": "Wole Soyinka", "label": "PER"}, {"text": "Nigerian", "label": "MI	{}

Tasks: 9 / 12 Submitted annotations: 0 Predictions: u

From a config to a running project

Defining the config is only the first step; loading data and labeling it is the rest. The short walkthrough below shows the full loop, creating a project, uploading a CSV of tasks, choosing the labeling template, and annotating every task:



 [Cite this page](#)  5 min read

CONTRIBUTOR  (<https://github.com/abumafriim>)

Training and Guidelines

Annotators are only as consistent as the instructions they are given and the training that aligns them. Much of what looks like annotator disagreement is really guideline disagreement, and most of that is fixable before the main task starts. This page is about getting the guidelines and the training right early, while it is still cheap.

Write guidelines around examples, not definitions

A definition tells an annotator what a label means. An example shows them how to apply it. Good guidelines do both and lean on the examples. For each label, give clear positive examples, give negative examples that mark the boundary against neighbouring labels, and include the edge cases that would otherwise be decided inconsistently by each annotator alone. Say explicitly how to treat uncertain, mixed, or "none of the above" items, because those are where agreement quietly breaks down. African-language guidelines carry an extra burden that English ones rarely face. They have to say how annotators should treat dialectal variation, code-switching between the target language and a colonial language, inconsistent or competing orthographies, and culturally specific expressions whose meaning a non-member would miss. Decide these cases once, in the guideline, rather than thousands of times, differently, in the data.



TEMPLATE: ANNOTATION GUIDELINES

A fill-in guidelines document modelled on MasakhaNER 2 and AfriSenti. It covers definitions, a step-by-step decision procedure, the label set, orthography and diacritics, agreement targets, adjudication, and a change log.

[Open the annotation guidelines template.](#)

Train and calibrate until annotators converge

Training turns a written guideline into shared judgment, so run it before annotation starts and give it structure rather than leaving it to a read-through. The Masakhane named-entity work offers a concrete model. For MasakhaNER 2.0, each language had a coordinator who first annotated a set of sentences themselves and was trained across two online workshops, and who then trained a team of three native-speaker annotators for that language (Adelani et al., 2022). Calibration rounds apply the same logic at a smaller scale: have several annotators label the same sample, compare their labels, discuss the disagreements openly, and repeat until agreement reaches an acceptable level before the full task begins. AfriSenti reached inter-annotator agreement above 0.70 on sentiment by combining native speakers with clear guidelines and exactly this kind of calibration (Muhammad et al., 2023). Agreement reached this way is earned, not assumed.

Pilot, then iterate the design

A pilot is a small dry run of the whole task, and its purpose is to find what is broken while fixing it is still cheap. Annotate a small but representative subset with your real annotators and your real tool, then look for the unclear instructions, the labels that collide, and the items that take far longer than expected. Measure consistency and difficulty, collect the annotators' own feedback on what confused them, and feed all of it back into the guidelines, the label set, and the workflow. Expect to revise. A pilot that changes nothing usually means nobody looked hard enough.

Prove the design with a minimum viable dataset

Before committing to full-scale annotation, build a minimum viable dataset, a small and representative slice annotated end to end. It validates the things that are expensive to get wrong at scale: whether the label schema actually covers the data, whether the workflow and tool hold up, and whether the per-item rate makes the full project affordable. Treat its results as a decision point. If the minimum viable dataset comes out clean and consistent, scale up. If it does not, redesign the task now, rather than after ten thousand items have inherited the flaw.

 [Cite this page](#)  3 min read

CONTRIBUTORS



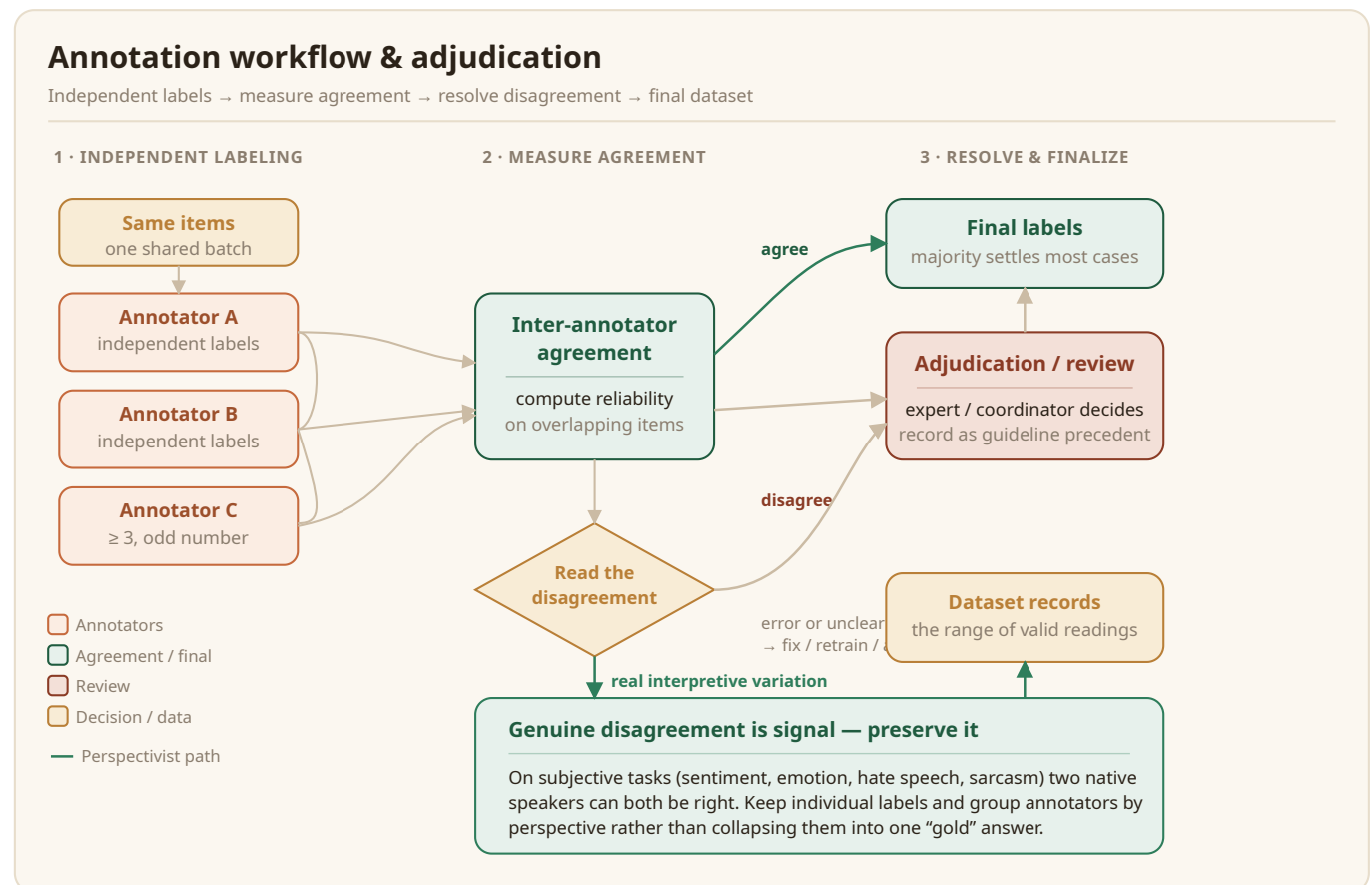
(<https://github.com/abumafrim>)



(<https://github.com/shmuhammadd>)

Workflow and Adjudication

Once the task is designed and annotators are trained, the workflow decides how labels are produced, checked, and finalised. The central questions are how many people label each item, how work is assigned, and what happens when they disagree.



Use multiple annotators per item

A single annotator's label is a single opinion, and you cannot tell a confident correct call from a confident mistake. Assigning each item to several annotators lets you measure reliability and resolve errors. The common practice for African-language datasets is at least three annotators per item, with an odd number so that a simple majority can settle most cases. The Masakhane named-entity datasets used three native speakers per language under a coordinator, which is enough to catch individual slips while staying affordable for a volunteer team (Adelani et al., 2022). More annotators raise reliability and cost together, so the right number is a budget decision as much as a statistical one.

Assign work with deliberate redundancy

Decide on purpose how items overlap between annotators. Some overlap is essential, because the items that more than one person labels are what let you compute agreement and spot annotators who have drifted. Mixing a small set of trusted, pre-labelled "gold" items into the stream, without flagging them, gives a continuous read on whether each

annotator is still applying the guideline correctly. Plan the assignment so that every item receives its required number of independent labels and so that no annotator reviews their own work, since self-review hides the very errors review exists to catch.

Resolve disagreement, but read it first

When annotators disagree, the usual resolution is a majority vote, with an expert or the language coordinator adjudicating the cases a vote cannot settle. Record those adjudication decisions, because they become precedent for the guideline and material for training the next round of annotators.

Before treating disagreement as something to erase, though, read what it is telling you. Not all disagreement is error. It can come from an ambiguous item, from a gap in the guideline, from a genuine mistake, or from legitimate differences in how people interpret subjective and culturally loaded content (Plank, 2022). That distinction matters a great deal for African data, where many of the most valuable tasks are subjective by nature. Sentiment, emotion, hate speech, and sarcasm all depend on dialect, region, and cultural context, and two native speakers from different communities can both be right. When disagreement traces to an unclear guideline or a genuine error, fix the guideline or retrain. When it reflects real interpretive variation, consider preserving the individual annotators' labels rather than collapsing them into one "gold" answer, so that the dataset records the range of valid readings instead of flattening minority perspectives out of existence. Recent African work goes further than simply keeping the labels: grouping annotators by how they agree, rather than reducing them to a majority vote, models these perspectives directly and improves performance on subjective tasks like sentiment, emotion, and hate speech across many languages (Belay et al., 2026). A label that hides genuine disagreement is less honest than one that shows it.

 [Cite this page](#)  3 min read

CONTRIBUTOR  (<https://github.com/abumafrim>)

Inclusive and Bias-Aware Annotation

Annotation is a human judgment, and human judgments carry the background of the person making them. Who annotates a dataset shapes what the dataset says, so inclusion and bias-awareness are not optional extras. They are part of getting the data right.

Why annotator background shapes the data

An annotator reads every item through their own language variety, region, gender, age, and lived experience. For a factual task that influence is small, but for the subjective tasks that dominate African-language work, such as sentiment, offensiveness, emotion, and sarcasm, it is decisive. A phrase that is ordinary in one community can read as an insult in another, and a tone that one generation hears as respectful another hears as cold. If a single demographic labels the whole dataset, its particular reading becomes the official "truth," and every model trained on the data inherits that narrowness. Diversity among annotators is therefore not only a matter of fairness. It is what makes the labels representative of the speakers the data is meant to serve.

Recruit for diversity, deliberately

Build the annotator pool to reflect the community whose language it is, across gender, age, and especially dialect and region. Avoid leaning on whichever group is easiest to reach, such as university students in the capital, when the language is spoken far more widely than that. Native-speaker involvement is the baseline, because native speakers catch the dialectal and idiomatic detail that outsiders miss, and the strongest African datasets have been built by recruiting native speakers from across the relevant communities rather than from one convenient pool ([Adelani et al., 2022](#); [Muhammad et al., 2023](#)).

Record annotator context, with consent

Where it is ethical and consented, record basic, anonymised information about each annotator, such as language background, region, and relevant expertise. This metadata serves two purposes. It lets you check after the fact whether labels split along demographic lines, which is how you detect bias rather than guess at it. It also makes a perspectivist approach possible, since you can only preserve and study different readings if you know whose reading each label is. Collect only what you need, anonymise it, and treat it under the same consent and privacy terms as the rest of the data (see [Data Governance](#)).

Build awareness, and involve the community

Train annotators to notice their own cultural and personal biases, using paired examples of biased and even-handed labelling, and reinforce that the goal is to apply the guideline consistently rather than to impose a personal verdict. Bias-awareness is not a one-off slide. It is a habit kept alive through calibration and feedback. Underneath all of this sits community participation. Engaging the communities whose language is being annotated, drawing on native-speaker knowledge, and respecting community norms is both a quality practice and an ethical one, and it follows directly from the principle that a community should hold authority over data about it ([Carroll et al., 2020](#)). Inclusive annotation and good governance are the same commitment seen from two angles.

 [Cite this page](#)  3 min read

CONTRIBUTOR  (<https://github.com/abumafrim>)

Annotation guidelines template

Last reviewed: 2026-07-07.

A task-agnostic starting point for a new corpus's annotation guidelines document, modelled on the structure MasakhaNER 2 ([Adelani et al., 2022](#)) and AfriSenti ([Muhammad et al., 2023](#)) use. Fork this file, rename it, drop your task in, and adapt the sections. The section structure is what matters: it is the same structure the playbook's Case Studies chapter compares projects against.

How to use this template

1. Copy everything below the divider into a new file (`docs/annotation-guidelines.md` in your project repo).
2. Replace every `[BRACKETED FIELD]` with a real answer.
3. Delete task-inappropriate sections and add a one-line `Not applicable: [why]` note in place of what you removed.
4. Version this document. Annotation guidelines drift; a guidelines document without a version and a change log is untrustworthy.
5. Store the guidelines alongside the corpus so future readers can verify what an annotator was actually told.

Motivated by the [Annotation Design](#) chapter and by every corpus retrospective in the [Case Studies](#) chapter that names annotation-guideline drift as a source of quality problems.

[PROJECT NAME] annotation guidelines

Version: [X.Y] · **Date:** [YYYY-MM-DD] · **Language(s):** [LANGUAGES] · **Task:** [TASK]

1. What we are annotating and why

- **The task in one sentence:** [E.g., "Classify each Twitter post in Hausa, Igbo, or Yoruba as positive, negative, or neutral sentiment."]
- **What the annotated corpus will be used for:** [Training a specific model, benchmark release, downstream deployment. Be specific; annotators annotate differently when they know the use case.]
- **What the corpus is NOT for:** [Avoid uses the community has agreed on. Explicit here helps the annotator make judgement calls at the margin.]
- **Reference project(s) this is modelled on:** [E.g., "MasakhaNER 2 guidelines, adapted for [DOMAIN]."]

2. The team

- **Lead annotator:** [NAME + short bio]
- **Native-speaker annotators:** [NAMES, or "N annotators, native speakers of [LANGUAGE], recruited via [CHANNEL]."]
- **Adjudicator:** [The person who resolves disagreements. Usually the lead annotator; can be someone else.]
- **Linguistic consultant, if any:** [NAME + role]
- **Reviewer / auditor:** [The person who periodically checks annotation quality without doing annotation themselves.]

- **How annotators were compensated:** [Rate + benchmark. Fair-rate community-anchored rates are part of the [core principles](#); state what you paid.]

3. Definitions

Every term the annotator uses is defined here, in one place, in plain language. **This is the most-important section.** Guideline drift usually starts with two annotators reading the same term and understanding it differently.

- **[TERM ONE]:** [Definition + one positive example + one negative example ("this is NOT [TERM]") for contrast.]
- **[TERM TWO]:** [Same shape.]
- **[TERM THREE]:** [Same shape.]

Add every term the annotator will encounter. Cross-reference to definitions used in the reference project (MasakhaNER 2, AfriSenti, LAFAND-MT) explicitly, noting any differences.

4. The annotation itself: decision procedure

Written as a step-by-step procedure the annotator can follow for a single example, from opening the labelling tool to submitting the annotation.

1. **Read the example fully before assigning any label.** [Especially important for classification tasks, since snap judgements based on the first few words are the main source of low-quality labels.]
2. **Check the context.** [If the example is a tweet, what is the thread? If the example is a sentence in a document, what is the surrounding paragraph? Instructions on how much context to consider.]
3. **Assign the label using the decision tree below.**
4. **If uncertain, mark the example as UNCERTAIN with a short note.** [Uncertain examples go to the adjudicator; do not force a label when unsure.]
5. **Submit and move on. Do not revise earlier annotations after the fact** unless a guideline change requires re-annotation.

4.a Decision tree

```
Is [DIAGNOSTIC QUESTION 1]?
├─ Yes → label = [LABEL 1]
└─ No: is [DIAGNOSTIC QUESTION 2]?
    ├─ Yes → label = [LABEL 2]
    └─ No: mark UNCERTAIN with a note explaining the ambiguity.
```

Replace with your task's diagnostic questions. The tree should be depth ≤ 4 ; deeper trees indicate the label set is not well-defined.

4.b Corner cases

The specific patterns that break the decision tree. Every case listed here is a case at least one annotator got wrong in the pilot round; write them down after the pilot, not before.

- **[CORNER CASE 1]:** [Example + correct label + reasoning.]

- **[CORNER CASE 2]:** [Same shape.]

5. Label set

If the task has more than three labels, name them here explicitly with examples.

Label	Definition	One positive example	One negative example
[LABEL 1]	[DEFINITION]	[EXAMPLE]	[WHAT LOOKS LIKE BUT ISN'T]
[LABEL 2]	[DEFINITION]	[EXAMPLE]	[WHAT LOOKS LIKE BUT ISN'T]
UNCERTAIN	Ambiguous; adjudicator will resolve.	n/a	n/a

For sequence-labelling tasks (NER, POS), replace with the tag inventory and the span-boundary rules; see the specific per-tag guidance in [MasakhaNER 2](#) as the reference.

6. Orthography, script, and diacritics

Load-bearing for African languages. Guidelines that skip this section produce corpora that cannot be evaluated consistently.

- **Script(s) used:** [E.g., "Latin only. Ajami content is out of scope for this corpus."]
- **Orthographic convention:** [Which spelling standard applies. If the language has more than one, name the one the corpus follows and cite the reference.]
- **Diacritic handling:** [Preserve fully, normalise to NFC, strip, with the reasoning.]
- **Handling of code-switched or mixed-script examples:** [Include, exclude, mark separately.]

7. Inter-annotator agreement

- **Metric used:** [Cohen's kappa for two annotators, Krippendorff's alpha for more, span-level F1 for sequence tagging. See the [evaluation-script template](#) for the metric implementations.]
- **Target level:** [E.g., " $\kappa \geq 0.7$ by the end of the second round"; state what "acceptable" means for this project.]
- **Cadence:** [How often IAA is measured: after every N examples per annotator, every round, etc.]
- **What happens when IAA is below target:** [Additional training, guideline revision, annotator recalibration. The response is a design choice; state it.]

8. Adjudication

- **Who adjudicates:** [Named person / role.]
- **When adjudication runs:** [Every UNCERTAIN mark? Every annotator disagreement? Sampled review?]
- **Decision authority:** [Final say on labels; state whether adjudicator decisions can be over-ruled by the lead annotator or the community reviewer.]
- **Guideline updates from adjudication:** [If adjudication reveals a systematic issue, when and how the guidelines are revised.]

9. Change log

Every version of these guidelines is recorded, dated, and briefly annotated. Guideline changes downstream of pilot annotation are normal; unrecorded changes are the problem.

Version	Date	Change	Impact on prior annotations
1.0	[YYYY-MM-DD]	Initial.	None
[NEXT]	[YYYY-MM-DD]	[CHANGE]	[Re-annotate? Discard? Keep as-is?]

10. Contact and community

- **Guidelines maintainer:** [NAME + contact]
- **How annotators raise concerns:** [Channel + response time commitment. See the [ombudsperson role](#) in the consent template; the same principle applies for guideline issues.]
- **Community reviewer, if any:** [NAME + role]

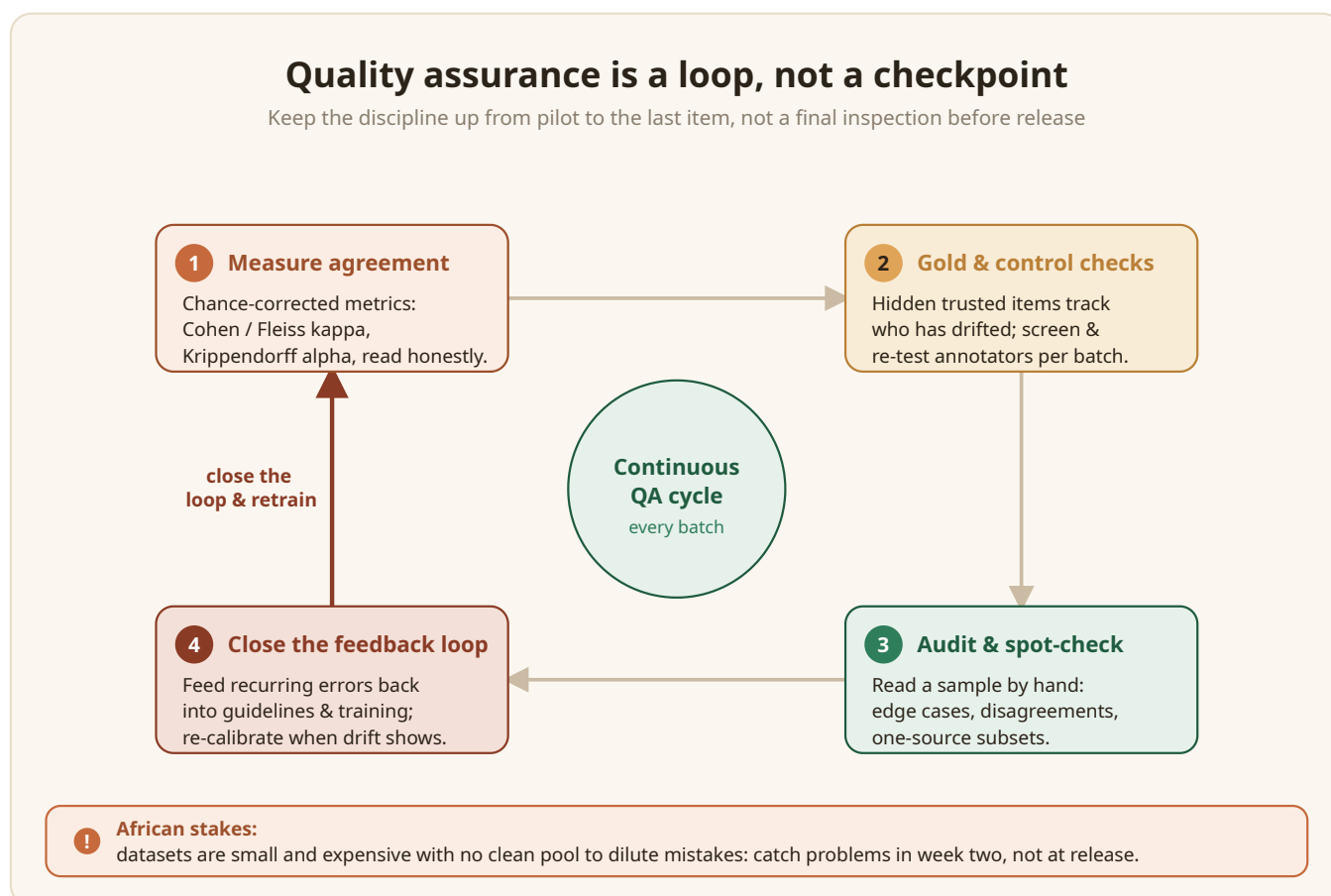
Contributor's note. If you have run an African-language NLP annotation project and have a version of this document that survived contact with real annotators, the highest-value contribution is a redlined comparison against this template with your project-specific additions clearly marked. Real annotation-guideline documents from real projects are more useful than generic prose.

 [Cite this page](#)  6 min read

CONTRIBUTORS  (<https://github.com/abumafri>)  (<https://github.com/shmuhammadd>)

Data Quality Assurance

Quality is not a final inspection you run before release. It is a discipline you keep up from the pilot to the last item. For low-resource African data the stakes are higher than usual, because every example is expensive to produce and there is no vast pool of clean data to dilute mistakes. When a large web crawl was audited, a striking share of the low-resource text turned out to be mislabelled, machine-translated, or not the target language at all (Kreutzer et al., 2022). A small, carefully assured dataset is worth far more than a large one nobody checked. This page covers the assurance side: measuring agreement, using gold data, auditing by hand, and closing the loop.



Measure inter-annotator agreement

When several people label the same items, agreement tells you whether the labels are reliable or just one person's opinion repeated. Raw percentage agreement is easy to read but misleading, because some agreement happens by chance, so use a chance-corrected measure: [Cohen's kappa](#) for two annotators, [Fleiss' kappa](#) for three or more, and [Krippendorff's alpha](#) when annotations are missing or the labels are ordinal. Production-grade work often targets an alpha above 0.8, though the right threshold depends on how subjective the task is. African datasets show the full range in practice. MasakhaNER reached high agreement by training annotators in workshops where they discussed disagreements (Adelani et al., 2022), AfriSenti held sentiment agreement above 0.70 (Muhammad et al., 2023), the Thiomi multimodal corpus kept Fleiss' kappa above 0.82 (Thiomi Dataset, 2026), and AfriHate reported Randolph's kappa

between 0.46 and 0.81 across its hate-speech datasets (Muhammad et al., 2025). Lower numbers are not automatically a failure: on genuinely subjective tasks they can reflect real interpretive variation rather than sloppiness (Plank, 2022). The worked formulas and code live in the task chapters; here the point is to choose the right metric and read it honestly.

Once you have chosen the metric, computing it on an exported annotation file is short. Tools using the Label Studio configuration format export annotations as JSON, so a task carries the labels every annotator gave it. The script below builds the annotator-by-item matrix from that export and reports the right statistic for the number of annotators, using established libraries rather than a hand-rolled formula.

```
import json
from collections import defaultdict

from sklearn.metrics import cohen_kappa_score  # pip install scikit-learn
from statsmodels.stats.inter_rater import fleiss_kappa, aggregate_raters

def load_labels(export_path: str, from_name: str = "sentiment") → dict:
    """From a Label Studio-format JSON export, return
    {task_id: {annotator: label}} for one labeling field."""
    by_task = defaultdict(dict)
    with open(export_path, encoding="utf-8") as f:
        tasks = json.load(f)
    for task in tasks:
        for ann in task.get("annotations", []):
            who = ann.get("completed_by", "unknown")
            for result in ann.get("result", []):
                if result.get("from_name") != from_name:
                    continue
                choice = result["value"]["choices"][0]  # single-choice task
                by_task[task["id"]][who] = choice
    return by_task

def agreement(by_task: dict) → None:
    # Keep only items every annotator labelled, so the comparison is fair.
    annotators = sorted({a for labels in by_task.values() for a in labels})
    shared = {t: l for t, l in by_task.items() if len(l) == len(annotators)}
    print(f"{len(annotators)} annotators, {len(shared)} commonly-labelled items")

    if len(annotators) == 2:
        a, b = annotators
        y1 = [shared[t][a] for t in shared]
        y2 = [shared[t][b] for t in shared]
        print(f"Cohen's kappa: {cohen_kappa_score(y1, y2):.3f}")
    else:
        # Fleiss' kappa expects an items x categories count table.
        table = [[shared[t][a] for a in annotators] for t in shared]
        counts, _ = aggregate_raters(table)
        print(f"Fleiss' kappa: {fleiss_kappa(counts):.3f}")

if __name__ == "__main__":
    agreement(load_labels("afriannotate_export.json"))
```

Reading agreement once at the end hides the drift this chapter warns about. The same export carries an annotation timestamp, so the same matrix can be split by batch or by time and scored separately, which is exactly how the Setswana sentiment corpus surfaced annotators slowly diverging across its eight batches (Abdulummin et al., 2026). When labels are ordinal or annotators skip items, swap these functions for Krippendorff's alpha (the `krippendorff` package), which handles both cases that kappa does not.

Use gold data and control checks

A gold standard is a set of items whose correct labels you already trust. Mixed into the annotation stream without being flagged, gold items give a continuous, per-annotator read on who is still applying the guideline and who has drifted, so you can retrain or remove an annotator before their errors spread through the data. Build the gold set carefully, because it becomes your reference for everything else, and remember that even gold is imperfect: audits of benchmark datasets find error rates from under one percent to over ten, depending on the task. Screen annotators the same way you check them, by having candidates pass a short annotation test before the main task and by re-showing a few items to measure each person's self-consistency.

Audit and spot-check by hand

Aggregate agreement can look healthy while something systemic is quietly wrong, so do not trust the numbers alone. Pull a sample and read it. Spend the time on the places problems hide: the edge cases, the items where annotators clustered into disagreement, and any subset from a single source or annotator. The web-crawl audit is the cautionary tale here, since a brief human look at a sample exposed whole languages that automated pipelines had passed as clean (Kreutzer et al., 2022). A few hours of careful reading catches failures that no metric will surface on its own.

Close the feedback loop

Quality assurance is a loop, not a checkpoint. Track agreement over the life of the project rather than only at the end, feed every recurring error back into the guidelines and the next round of training, and re-calibrate when annotators drift apart. Drift is real and measurable, and a single end-of-project agreement score can hide it. In a Setswana sentiment corpus, overall agreement looked healthy at $\kappa = 0.76$, yet items labelled within a minute of each other reached $\kappa = 0.98$ while items labelled more than a day apart fell to $\kappa = 0.65$, a clear sign that annotators slowly diverged across the eight batches of the project (Abdulmumin et al., 2026). The reason to monitor continuously is economic as well as statistical: a problem caught in week two costs a fraction of the same problem discovered at release, when it has already contaminated the whole dataset (Sambasivan et al., 2021).

 [Cite this page](#)  5 min read

CONTRIBUTOR  (<https://github.com/abumafrim>)

Data Quality Management

Assurance keeps individual labels honest. Management looks after the dataset as a whole: its balance, its noise, the patterns in its errors, and the way it changes over time. These are the dataset-level properties that decide whether a corpus is trustworthy and reusable.

Handle class imbalance

Low-resource datasets are rarely balanced. Hate speech is a small fraction of normal posts, a minority dialect is a small fraction of a national language, and rare entity types barely appear. Left alone, imbalance lets a model score well by ignoring the classes you care about most. There are three honest responses, and most projects combine them. Enrich rare classes at collection time through targeted or keyword-based sampling, as hate-speech and emotion projects do to find enough positive cases. Use stratified sampling so minority classes and subgroups are always present in the annotation set. And when imbalance is unavoidable, account for it in evaluation by reporting per-class results rather than a single accuracy that the majority class can dominate. Decide which of these you are relying on, and write it down, because an unstated sampling choice is itself a quality problem.

Detect noise and outliers

Noise is the default state of collected African-language text, especially anything scraped, where wrong-language passages, machine-translated filler, and duplicates are common ([Kreutzer et al., 2022](#)). Clean in layers. Run language identification to remove off-target text, using tools built for African languages such as AfroLID or GlotLID rather than general-purpose detectors that misread them. Remove exact and near-duplicates before splitting the data, so the same text cannot leak across train and test. Then look for the subtler outliers: items whose length, formatting, or content sits far outside the rest, or which a simple model confidently disagrees with, since both are useful flags for a human to inspect. The goal is not to delete everything unusual, because genuine dialectal variation can look like an outlier, but to surface it for a decision.

Build an error-analysis pipeline

When a model trained on the data makes mistakes, treat those mistakes as data about the dataset. A repeatable error-analysis pipeline groups errors into categories, looks for patterns by class, dialect, source, or annotator, and traces each pattern back to a root cause, whether that is an ambiguous guideline, a label that overlaps another, a noisy source, or a tooling quirk. The payoff is that most findings convert directly into fixes: a sharpened guideline, a merged or split label, a re-cleaned source. Error analysis is also where the line between a genuine model error and a genuine label error gets drawn, which feeds straight back into the assurance loop.

Version the dataset

A dataset is not finished at first release; it is corrected, extended, and re-split over time. Treat it like code. Give every release a version number, record what changed and why in a short changelog, and keep the raw collection and the processed versions separate so the original is never overwritten. Versioning is what makes results reproducible, lets a

downstream user cite the exact data they trained on, and allows you to roll back when a well-meant "fix" turns out to introduce a regression. Pair the version history with the dataset's documentation and provenance (see [Documentation](#) and [Data Governance](#)), so that what changed, who changed it, and under what terms are all answerable years later.

 [Cite this page](#)  3 min read

CONTRIBUTOR  (<https://github.com/abumafrim>)

Community Ecosystems

African NLP is, to an unusual degree, a community endeavour. The datasets, models, and benchmarks that move the field forward are built less by large institutions than by distributed networks of volunteers, students, and researchers who coordinate online across the continent. The Masakhane community captures the spirit in its name, which means "we build together" in isiZulu, and in its insistence that African language technology should be built by Africans, for Africans, with the community owning what it creates ([Masakhane](#)). Knowing this ecosystem is practical knowledge, because it is where you find collaborators, datasets, annotators, and funding.

The communities and what they do

The ecosystem has a recognisable shape. **Masakhane** is the pan-African hub, a grassroots community of thousands whose participatory model produced foundational datasets such as MasakhaNER and MAFAND ([Nekoto et al., 2020](#); [Adelani et al., 2022](#)). Around and within it sit **language- and country-focused groups** like HausaNLP, EthioNLP, GhanaNLP, and Digital Umuganda in Rwanda, which carry deep expertise in their specific languages. The **Deep Learning Indaba**, founded in 2017, is the community's annual meeting place and runs IndabaX events that had reached 47 countries by 2025, building the human capacity the field depends on ([Deep Learning Indaba](#)). **AI4D-Africa** funded and coordinated dataset creation through its African Language Program ([Siminyu et al., 2021](#)), **Zindi** hosts competitions and a large pool of African data scientists, **Lelapa AI** builds African-centred models and the Esethu framework, and **Lanfrica** maps the resources so they can be found ([Lanfrica](#)). This is not an exhaustive list, and it grows every year.

Academic and industry collaboration

Good African-language datasets increasingly come from partnerships that cross the academic, industry, and community lines. University centres such as the Data Science for Social Impact group at the University of Pretoria, the Maseno Centre for Applied AI in Kenya, and SADiLaR, South Africa's national language-resource infrastructure, bring linguistic depth and continuity. Companies and funders bring scale and money: African Next Voices paired community centres with Gates Foundation funding to record thousands of hours of speech ([African Next Voices, 2025](#)), and Lacuna Fund underwrites dataset creation across the continent ([Lacuna Fund, n.d.](#)). The healthiest of these partnerships keep the community in control of the data even when the money comes from outside, which is exactly what the [data governance](#) chapter argues for.

Contributing, and crediting contributors

A community resource is only as strong as the people who contribute to it, so make contribution easy and make credit real. Provide clear contributor guidelines that say what kinds of help are wanted, whether that is writing, reviewing, sharing a dataset, or translating, and how to get started. Lower the barrier to a first contribution, because many of the field's most valuable people began as students who fixed one small thing. And name contributors in datasheets, in publications, and in release notes, because recognition is both fair and the cheapest way to keep a volunteer community alive. The growth of African NLP over two decades, from a handful of papers a year to hundreds, has been driven precisely by this widening circle of contributors ([Belay et al., 2025](#)).

CONTRIBUTOR  (<https://github.com/abumafrim>)

Collaboration and Shared Tasks

Beyond standing communities, African NLP advances through organised collaboration: shared tasks, benchmarks, workshops, and open challenges that pool effort toward a common goal. For a field where no single group can cover hundreds of languages, these are the mechanism that turns scattered work into shared infrastructure.

Shared tasks and benchmarks

A shared task sets a common problem, a common dataset, and a common way of scoring, then invites everyone to take part. The effect multiplies effort: many teams build data and models for the same goal at once, and the field gets a benchmark to measure progress against afterward. African NLP has used this pattern repeatedly and well. AfriSenti became a SemEval shared task that produced sentiment datasets for fourteen languages and drew dozens of participating systems ([Muhammad et al., 2023](#)), the MasakhaNER efforts coordinated named-entity annotation across twenty languages ([Adelani et al., 2022](#)), and benchmarks like IrokoBench now give the community a shared yardstick for large language models on African languages. Contributing to a shared task is one of the most efficient ways to add a new language to the map, because you inherit the protocol, the tooling, and the company of others doing the same work.

Workshops and open challenges

Workshops and challenges are where the community gathers, plans, and recruits. The AfricaNLP workshop, held annually alongside major conferences, is the field's main venue for new African-language work, and the Deep Learning Indaba and its IndabaX events build the skills and relationships that make collaboration possible in the first place ([Deep Learning Indaba](#)). Open challenges turn this energy toward data directly. Zindi competitions and the AI4D African Language Dataset Challenge incentivised the creation of open datasets that did not exist before ([Siminyu et al., 2021](#)), and the Deep Learning Indaba's 2026 call for African datasets continues the pattern across text, speech, and vision. If you are starting out, joining one of these is often a better first move than working alone, because the community will help you build something that lasts.

 [Cite this page](#)  2 min read

CONTRIBUTOR  (<https://github.com/abumafrim>)

Text

Building text datasets for African languages. Every task below shares the same groundwork; these pages differ only in what is **labelled** and how it is **evaluated**.

Shared across text data

- Source selection (news, web, social media, books, religious & educational texts)
- Text cleaning & normalization (encoding, scripts, diacritics, punctuation)
- Language identification & code-switching handling
- Annotation guidelines & label schemas
- Annotator recruitment & inter-annotator agreement
- Metadata, dataset formatting & splits
- Licensing & toxic-content filtering

Tasks

- **Text classification & labeling** – (sentiment, emotion, hate speech, topic, intent)
- **Text generation, summarization & QA** – (open generation, summarization, question answering)
- **Machine translation & parallel corpora** – (parallel mining, alignment, human translation)
- **Sequence labeling** – (named-entity recognition, part-of-speech, chunking)

 [Cite this page](#)  1 min read

CONTRIBUTORS



(<https://github.com/abumafrim>)



(<https://github.com/shmuhammadd>)

Defining text classification tasks:

Text classification is a supervised NLP task in which a text is assigned one or more labels from a predefined label set. In this playbook, we focus on four common text classification tasks: sentiment analysis, emotion analysis, hate speech analysis, and topic classification. Although these tasks can overlap in practice, they differ in what they aim to capture: sentiment focuses on polarity, emotion focuses on affective state, hate speech focuses on harmful or discriminatory language

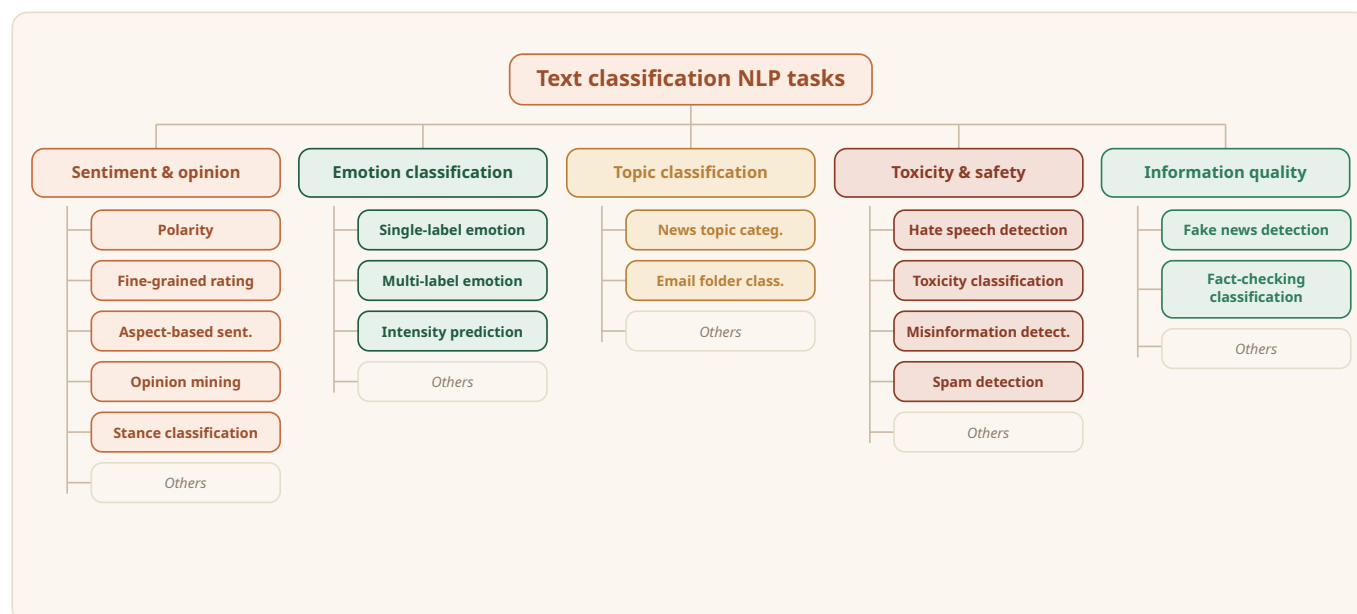
! SCOPE NOTE

This chapter is designed for dataset creation and annotation. It does not cover downstream model training in detail, but the annotated outputs can later be used for classification, retrieval, moderation, or analytics pipelines.

Task distinction

Below is a short definition of the common NLP tasks (the question asks and answers). The details of each task are discussed later.

- Sentiment analysis: Is the text positive, negative, neutral, or mixed?
- Emotion analysis: What emotion or emotions are expressed?
- Hate speech analysis: Does the text contain hateful, offensive, or discriminatory language, and who is targeted?
- Topic classification: What is the main theme or domain of the text?



! DEFINE TASKS

Define the annotation objective before collecting data. A dataset built for sentiment analysis should not be reused for hate speech or emotion analysis without revisiting the label schema and annotation guidelines.

CONTRIBUTORS



(<https://github.com/abumafri>)



(<https://github.com/shmuhammad>)



(<https://github.com/Tadesse-Destaw>)

Collecting & Preparing Data

This page walks through the data stages of a text-classification project end to end: choosing **where** the data comes from, **how** to select and sample it, how to **clean and standardise** it before annotation, and how to keep **quality** under control throughout.

Data sources

Data source is the place where we get the data (text, audio, image or any of the combinations) to be annotated. The best data source depends on the task, language, domain, and cultural context. Product reviews are often useful for sentiment analysis because they contain explicit evaluative language. Social media posts are especially useful for emotion analysis and hate speech analysis because they capture spontaneous expression, disagreement, and interactional language. Forums, blogs, and comment sections can provide longer and more context-rich texts, while survey responses can be useful when researchers need cleaner data or want to target a specific population.

Common Data Sources: Selecting data sources that are relevant, ethical, and representative is essential for building high-quality text classification datasets such as sentiment, emotion, and hate speech datasets. Common sources include social media platforms such as Twitter (X), Facebook, Reddit, YouTube, TikTok, Telegram, and WhatsApp, which provide rich and real-time user opinions but often contain noisy and informal language. Product review platforms such as Amazon typically offer clearer sentiment signals, while forums, blogs, and news comment sections provide diverse viewpoints and discussions. Researchers may also collect data through surveys or controlled studies, which generally produce cleaner but smaller datasets. Additionally, existing benchmark datasets can accelerate research and enable comparison with prior work, although they may not always align with the target domain, language, or cultural context.



Social media

Rich, real-time opinions, but noisy and informal. Key for hate speech & emotion tasks.

[Twitter \(X\)](#)
[Facebook](#)
[Reddit](#)
[YouTube](#)
[TikTok](#)
[Telegram](#)
[WhatsApp](#)


Product reviews

Clearer sentiment signals: structured opinions tied to a rated product.

[Amazon](#)
[Yelp](#)


Forums & comments

Diverse viewpoints across topics, useful for broad coverage.

[Blogs](#)
[News sites](#)
[Reddit](#)


Surveys & collected data

Controlled and cleaner data: smaller volumes but higher label quality.

[Questionnaires](#)
[Focus groups](#)
[Annotation tasks](#)


Existing datasets

Benchmark corpora for faster start; may not match your domain.

[Benchmark corpora](#)
[HuggingFace datasets](#)
[Published research data](#)

! TIPS

When selecting a source, prefer data that naturally contains the phenomenon of interest. For example, if the goal is to study offensive speech, choose platforms where interpersonal conflict or public debate is common. If the goal is to study sentiment, choose sources where opinions, reviews, and evaluations are frequent.

BENCHMARK DATASETS

Benchmark datasets are useful for comparison, but they may not reflect the target language variety, region, or social context. Always check whether the original data distribution matches your intended use case.

Collection & selection approaches

Data can be collected through APIs, web scraping (with permission), manual collection, or surveys, while preserving useful metadata such as source, time, language, and identifiers for future analysis. Data sources should be relevant to the target domain, language, and cultural context, with careful attention to dataset quality, class balance, and representativeness. Throughout the process, researchers must also address ethical and legal requirements, including privacy, consent, and compliance with platform policies. Data samples can be collected using one of the approaches below.

TIPS

Data collection should be guided by the annotation objective and the expected label distribution. For example, if a dataset is likely to contain too few hateful texts, keyword-based filtering or distant supervision can be used to enrich the sample. If the goal is to estimate natural distribution, random sampling or stratified sampling is more appropriate.

Keyword/Dictionary-based Selection: Select documents or sentences that contain one or more predefined keywords, phrases, or lexicon entries. For example, in hate speech detection, a list of commonly used hate-related or offensive terms in the target language can be compiled and used to identify potentially relevant texts. This method helps enrich the dataset with task-relevant examples while reducing the amount of irrelevant data. A widely used resource for English emotion-related keywords is the NRC Emotion Lexicon.

```
# Keyword / dictionary-based selection (e.g. hate-speech terms or NRC emotion lexicon)
import re, pandas as pd

lexicon = {"insult1", "slur2", "offensive3"} # build per target language
pattern = re.compile(r"\b(" + "|".join(map(re.escape, lexicon)) + r")\b", re.I)

df = pd.read_csv("corpus.csv")
df["matched"] = df["text"].str.contains(pattern)
selected = df[df["matched"]]
print(len(selected), "candidate texts enriched for the task")
```

PROS

- ✓ Highly targeted samples
- ✓ No model needed to select
- ✓ Interpretable & auditable

CONS

- ✗ Misses paraphrases & synonyms
- ✗ Selection bias toward keywords
- ✗ Requires domain knowledge

Location-based Selection: Is a data collection approach where texts are gathered based on the geographic location associated with users or posts. For example, when collecting social media data from X (formerly Twitter), researchers can filter posts originating from specific locations such as Ethiopia, Nigeria, or Kenya. This method is useful for studying regional language variation, local opinions, cultural expressions, or location-specific events, as it helps ensure that the collected data represents the target geographic area.

```
# Location-based filtering of collected posts (place / bounding box)
TARGET = {"Ethiopia", "Nigeria", "Kenya"}

def in_region(post):
    place = (post.get("place") or {}).get("country")
    return place in TARGET

geo = [p for p in posts if in_region(p)]
print(len(geo), "geotagged posts (note: usually only ~1-3% are geotagged)")
```

PROS

- ✓ Enables dialect / regional studies
- ✓ Precise targeting for geo tasks
- ✓ Scales easily with bounding boxes

CONS

- ✗ Only ~1–3% of tweets are geotagged
- ✗ Location ≠ author's language region
- ✗ Platform-dependent metadata

Distant supervision: Is a method for automatically creating labeled training data by using existing knowledge sources instead of manual annotation. For example, for emotion classification, social media posts containing hashtags such as **#happy**, **#joy**, or **#sad** can be automatically labeled with the corresponding emotions. This approach enables the creation of large training datasets quickly and cheaply, although some automatically assigned labels may be incorrect or noisy.

```
# Distant supervision: weak labels from emotion hashtags
HASHTAG_EMOTION = {"happy": "joy", "joy": "joy", "sad": "sadness", "angry": "anger"}

def weak_label(text):
    tags = re.findall(r"#(\w+)", text.lower())
    for t in tags:
        if t in HASHTAG_EMOTION:
            return HASHTAG_EMOTION[t]
    return None

df["weak"] = df["text"].apply(weak_label)
```

PROS

- ✓ Millions of examples, no manual labels
- ✓ Enables training without annotation budget
- ✓ Can bootstrap active-learning cycles

CONS

- ✗ Label noise can hurt model quality
- ✗ Assumes signal aligns with task
- ✗ Needs noise-aware training strategy

Random Sampling: Select items uniformly at random from the corpus with no targeting criteria. This is the baseline for any annotation project, ensuring an unbiased estimate of the true corpus-level label distribution.

```
# Random baseline vs. stratified sampling that preserves class balance
from sklearn.model_selection import train_test_split

# Pure random (unbiased estimate of the true distribution)
rand = df.sample(n=2000, random_state=42)

# Stratified by a known column (guarantees rare classes appear)
strat, _ = train_test_split(df, train_size=2000,
                           stratify=df["weak"], random_state=42)
print(strat["weak"].value_counts(normalize=True))
```

PROS

- ✓ Zero selection bias: most representative
- ✓ Simple to implement and audit
- ✓ Best for estimating true distributions

CONS

- ✗ Rare classes underrepresented
- ✗ Wastes budget on uninformative samples
- ✗ Inefficient for imbalanced tasks

Active Learning Method: Iteratively train a model on a small seed set, then use the model's uncertainty to select the most informative unlabelled examples for human annotation next. Maximizes annotation return on investment by labeling only where the model is confused.

```
# Active learning: label where the model is least confident (uncertainty sampling)
import numpy as np

probs = model.predict_proba(unlabeled_X) # shape (N, n_classes)
margin = np.sort(probs, axis=1)[: , -1] - np.sort(probs, axis=1)[: , -2]
to_label = np.argsort(margin)[:100] # 100 most ambiguous items
print("Send these to annotators next:", to_label)
```

PROS

- ✓ Up to 70% fewer labels for same accuracy
- ✓ Naturally surfaces hard edge cases
- ✓ Improves model and data simultaneously

CONS

- ✗ Requires annotation loop infrastructure
- ✗ Bias toward uncertain decision boundary
- ✗ Cold-start: needs an initial labelled seed

Stratified Sampling: Divide the corpus into strata (subgroups by class, source, time period, or demographic) and sample proportionally or equally from each. Ensures minority classes and subgroups are always represented in the annotation set.

PROS

- ✓ Guarantees coverage of rare classes
- ✓ Controls for known confounders
- ✓ Produces balanced training sets

CONS

- ✗ Requires pre-existing strata metadata
- ✗ Equal sampling distorts true distribution
- ✗ Strata boundaries can be arbitrary

Collection strategy guide

- Use keyword/dictionary-based selection when you need to target specific phenomena such as hate-related expressions or emotion lexicons.
- Use location-based selection when the research question involves dialect, region, or location-specific discourse.
- Use distant supervision when you need large amounts of weakly labeled data and can tolerate some label noise.
- Use random sampling when you want unbiased estimates of class prevalence.
- Use active learning when annotation budget is limited and model uncertainty can help choose informative examples.
- Use stratified sampling when you want to preserve balance across classes, sources, time periods, or demographic groups. Any combination of the above also works well to filter quality data.

Comparisons of data selection methods

Method	Cost	Label noise	Bias risk	Best for
Keyword / lexicon	Low	n/a	High (toward keywords)	Hate speech, rare phenomena
Location-based	Low	n/a	Medium	Dialect / regional studies
Distant supervision	Low	High	Medium	Large weakly-labelled emotion sets
Random sampling	Low	n/a	None	Estimating true distribution
Stratified	Medium	n/a	Low	Guaranteeing rare-class coverage
Active learning	High setup	n/a	Toward decision boundary	Maximising labels-per-dollar

! TIPS

Always store metadata such as source, timestamp, language, and collection method. This makes later analysis, error inspection, and dataset documentation much easier.

Processing & sampling

After collection, texts should be cleaned and standardized before annotation. Remove obvious noise such as HTML tags, duplicate items, extra whitespace, URLs, and non-textual tokens. Apply language identification to filter out non-target languages, and remove exact or near duplicates to reduce annotation waste and leakage across splits.

Once the data source has been identified, several preprocessing and sampling steps are required to ensure the quality and representativeness of the dataset. First, texts should be carefully selected to reflect the diversity of the target population and minimize potential biases. For tasks such as hate speech and emotion analysis, keyword-based filtering can be useful for identifying relevant content. Data cleaning involves removing irrelevant elements such as HTML tags, URLs, special characters, and excessive whitespace using text-processing tools. Applying language identification and de-duplication helps eliminate non-target language and repeated content. The overall dataset size should be determined based on factors such as research objectives, available resources, annotation budget, human capacity, and project timelines.

! TIPS

Keep a copy of the raw data and a separate processed version. Never overwrite the original collection, because later decisions may need to be audited or reversed.

01 · Data selection

Representative sampling

Keyword-based selection for bias control across hate speech & emotion tasks

02 · Data cleaning

Remove noise

Strip HTML tags, whitespace; automated via regex & text libraries

03 · Language ID

Uniform language

Ensure consistency across multilingual & mixed-language sources

04 · De-duplication

Eliminate repeats

Remove duplicate entries that skew model training & evaluation

05 · Dataset sizing

Scope the corpus

Balance budget, time, human resources & available data volume

06 · Quality check

Validate & review

Spot-check samples, inter-annotator agreement, label consistency



processed corpus ready for sampling



Sample selection

Draw representative, stratified samples from the final cleaned corpus for annotation & modelling



feeds into classification tasks



Sentiment analysis



Hate speech detection



Emotion analysis



Topic classification



Others

! TIPS

For hate speech and emotion tasks, keyword filtering can help find likely positive cases, but it should never replace careful annotation, because keywords often miss indirect, ironic, or context-dependent expressions.

Cleaning pipeline example

```
import re, hashlib, html

def clean(text):
    text = html.unescape(text)
    text = re.sub(r"https?:\/\/\S+", " URL ", text)      # keep a placeholder, don't delete
    text = re.sub(r"@w+", " @USER ", text)             # anonymise mentions
    text = re.sub(r"\s+", " ", text).strip()            # collapse whitespace
    return text                                          # NOTE: emojis intentionally kept

def dedup_key(text):
    return hashlib.md5(text.lower().encode()).hexdigest()

df["text"] = df["raw"].apply(clean)
df = df.drop_duplicates(subset=df["text"].apply(dedup_key)) # dedup BEFORE splitting
```

Quality control

Quality control should begin before full-scale annotation and continue throughout the process. Start with a pilot phase, then refine the guidelines, then monitor annotator performance using gold-standard items, disagreement review, and periodic feedback.

A practical quality-control workflow is:

Pilot annotation → guideline revision → main annotation → gold checks → disagreement review → adjudication → final dataset release.

Data quality control ensures that your sentiment labels are accurate, consistent, and reliable.

- Clear annotation guidelines: Provide precise definitions of sentiment classes (positive, negative, neutral) with examples, including tricky cases like sarcasm, negation, and mixed opinions. This reduces confusion and inconsistency.
- Multiple annotators & agreement: Assign each item to at least 2–3 annotators and measure how much they agree (e.g., using Kappa). High agreement indicates reliable labels.
- Gold-standard checks: Include a small set of pre-labeled (trusted) examples during annotation to evaluate annotator performance continuously.
- Disagreement resolution: Use majority voting or expert review to finalize labels when annotators disagree.
- Data cleaning: Remove duplicates, irrelevant content, spam, and noisy text to improve overall dataset quality.
- Class balance: Check that sentiment categories are not overly skewed (e.g., too many positives), or account for imbalance during modeling.
- Ongoing monitoring: Track annotator behavior over time to detect inconsistency or fatigue and take corrective action.

In short: combine good guidelines, multiple reviews, and continuous checks to maintain high-quality sentiment data.

Quality-control policy

- Use at least three annotators per item, and preferably more when feasible.
- Include gold-standard examples to monitor accuracy over time.
- Review items with persistent disagreement.
- Remove annotators whose responses are random, careless, or systematically inconsistent.

- Record adjudication decisions for transparency.

Example how to make majority voting:

```
from collections import Counter
def majority_vote(labels):
    return Counter(labels).most_common(1)[0][0]
```

 [Cite this page](#)  9 min read

CONTRIBUTORS  (<https://github.com/abumafrim>)  (<https://github.com/shmuhammadd>)

Annotation

This page covers the human side of building a text-classification dataset: **who** annotates (recruitment and well-being), **with what** (tools), and how to keep labels trustworthy (**agreement** and **quality control**).

Choosing annotators

Annotator quality is more important than annotator quantity. Recruit annotators who are fluent in the target language, familiar with the cultural context, and, when needed, knowledgeable about the topic domain. For emotion and hate speech tasks, it is especially important that annotators understand informal language, sarcasm, euphemism, and context-dependent expressions.

Effective annotation relies more on the quality and consistency of annotators than on their number. Annotators should be fluent in the target language, familiar with the relevant cultural context, and, when necessary, possess domain-specific knowledge. They should also be detail-oriented and able to apply annotation guidelines consistently to ensure reliable, high-quality labels.

WHO TO SELECT



Fluent / native speaker

Strong command of the target language is non-negotiable



Culturally aware

Understands slang, sarcasm, and local context



Domain-aware

Subject expertise required for specialized tasks



Detail-oriented

Consistent and meticulous across annotation sessions

KEY CONSIDERATIONS



Screening test

Small labeled sample; retain only high-performing candidates



Training

Clear guidelines with worked examples covering tricky edge cases



Multiple annotators

At least 2–3 annotators per item for reliable label coverage



Agreement check

Monitor consistency using inter-rater metrics (e.g., Cohen's Kappa)



Quality control

Embed gold-standard items; track performance over time



Bias & ethics

Minimize subjective bias; ensure fair and equitable treatment

! TIPS

Use a short qualification round before the main task. The goal is not only to select competent annotators, but also to identify whether the guidelines are clear enough to be applied consistently.

Before annotation begins, screen annotators for:

- Language proficiency.
- Domain familiarity.
- Ability to follow instructions carefully.

- Comfort with sensitive or harmful content.
- Availability for training and calibration.

Annotator safety & well-being

Protecting annotator well-being is essential, particularly when working with harmful, offensive, or emotionally distressing content. Annotators should be informed about potential risks before participation, allowed to opt out of sensitive tasks, and given the freedom to skip items or withdraw without penalty. Exposure to harmful content should be carefully managed through content filtering, workload limits, regular breaks, and task rotation. Projects should provide appropriate training, clear safety protocols, and access to psychological support resources when needed. Continuous monitoring of annotator well-being, respectful communication, protection of privacy, fair compensation, and adherence to ethical and legal standards are also critical for maintaining a safe and sustainable annotation environment.

Protecting annotators through informed consent, controlled exposure, breaks, and support systems improves both well-being and annotation quality.



For any data annotation such as hate speech, we should create a content warning + opt-in (show before any harmful-content task) like below.

CONTENT WARNING

This task contains hate speech, offensive, or distressing material. • You may skip any item, take breaks at any time, or withdraw with no penalty. • If you feel distressed, stop and contact the support team. [] I understand and choose to participate in this task.

Annotation tools

Text classification data can be annotated using a range of tools, from managed crowdsourcing platforms to self-hosted open-source systems. The choice of tool should depend on the task design, dataset size, number of annotators, required turnaround time, and the availability of qualified annotators.

There is no single best tool. When selecting an annotation tool, the right choice depends on several factors:

- Data sensitivity and privacy requirements.
- Size of the dataset and expected annotation volume.
- Number of annotators and required collaboration features.
- Need for quality assurance, audit trails, and reviewer workflows.
- Support for the annotation schema and task type.
- Availability of self-hosting, access control, and export options.
- Cost, ease of setup, and long-term maintainability.

Based on how annotators are sourced and where the data is hosted, annotation tools can be broadly grouped into two categories: crowdsourcing platforms and in-house (self-hosted) tools. The most common options for text classification are described below.

Crowdsourcing platforms

Crowdsourcing platforms give you access to a large, on-demand pool of remote annotators who are recruited and paid through the platform. Annotators are typically selected based on profile attributes (language, location, demographics, prior approval rating, or qualification tests) rather than being known to you personally. This makes crowdsourcing well suited to large volumes of data and widely spoken languages, where a broad annotator pool is readily available.

Common platforms include:

- Toloka AI: <https://toloka.ai> ↗
- Amazon Mechanical Turk (MTurk): <https://www.mturk.com> ↗
- Prolific: <https://www.prolific.com> ↗
- Appen: <https://www.appen.com> ↗
- Label Studio Enterprise: <https://labelstud.io> ↗

Many of these platforms support multiple data modalities (text, image, audio, video) and increasingly offer AI-assisted features such as pre-labeling, model-in-the-loop suggestions, and automated quality checks.

Trade-offs. Crowdsourcing scales easily and reduces recruitment overhead, but it offers less direct control over annotators and may make it difficult to source native speakers of low-resource languages or specific dialects. It also requires careful task design, qualification filters, and it raises data-privacy considerations because the data is sent to an external platform.

In-house (self-hosted) tools

In-house tools are typically open-source applications that can be customized, deployed, and hosted on your own machine or server. You create accounts for a hand-picked set of annotators (often colleagues, domain experts, or recruited native speakers), giving you full control over who labels the data and where the data lives. This category is preferred for sensitive data, specialized domains, and low-resource languages, where annotator expertise matters more than raw scale.

Common self-hosted tools include:

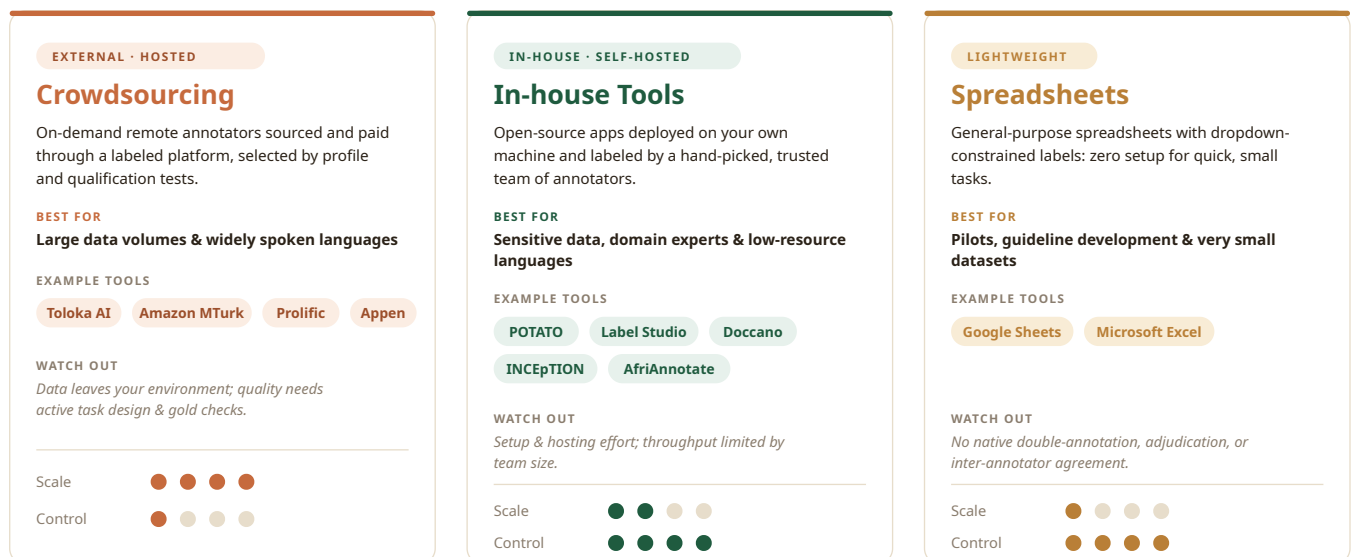
- AfriAnnotate, the playbook's companion annotation tool, built on the Label Studio configuration format: <https://afriannotate.waraka.org/> ↗
- POTATO (Portable Text Annotation Tool): <https://github.com/davidjurgens/potato> ↗
- Label Studio (open-source edition): <https://labelstud.io> ↗
- Doccano: <https://github.com/doccano/doccano> ↗
- INCEpTION (the successor to WebAnno): <https://inception-project.github.io> ↗
- brat: <https://brat.nlplab.org> ↗

Trade-offs. Self-hosted tools keep data fully under your control and can be tailored to bespoke label schemes and guidelines, but they require setup, hosting, and maintenance effort, and the annotation throughput is limited by the size of your recruited team.

Lightweight tools for small datasets

For small annotation efforts, a dedicated platform may be unnecessary. Spreadsheets (Google Sheets or Microsoft Excel) are a practical, zero-setup option: one column holds the text, and one or more columns capture the label(s), with data validation or dropdown lists used to constrain inputs to the allowed label set. Spreadsheets are easy to share and require no technical onboarding, which makes them convenient for pilot studies, guideline development, and very small in-house tasks.

However, these tools lack the quality-control and management features; they are not recommended beyond small or exploratory datasets.



Scale = data volume the approach supports | Control = direct oversight of who annotates and where data lives

The annotation tool should match the size and sensitivity of the project. For small pilot studies, spreadsheets may be enough. For medium-sized projects that require labeling workflows, Doccano, AfriAnnotate, or INCEpTION are often suitable. For large projects with distributed workers, crowdsourcing platforms can offer scale, but they require stronger quality control and privacy planning.

A labeling configuration

Whichever self-hosted tool you choose, the labeling task is defined by a small configuration. The examples here use the Label Studio configuration format, an open XML format supported by several tools (AfriAnnotate among them), so the same config is portable. The design points above (constrained inputs, keyboard shortcuts, an explicit skip route) are expressed directly in it. A single-label sentiment task looks like this:

```
<View>
  <View style="background:#FBF7F0; border:1px solid #E7DDCB; border-radius:8px; padding:14px 16px; margin-bottom:18px;">
    <Text name="text" value="$text"/>
  </View>
  <Choices name="sentiment" toName="text" choice="single" required="true">
    <Choice value="Positive" hotkey="1"/>
    <Choice value="Neutral" hotkey="2"/>
    <Choice value="Negative" hotkey="3"/>
  </Choices>
</View>
```

Emotion analysis is usually multi-label, since one sentence can carry several emotions at once, and each emotion present has its own intensity (see [Emotion Analysis](#)). The cleanest way to capture both is one intensity rating per emotion, so each instance gets several single annotations at once. A rating of zero stars means the emotion is absent:

```

<View>
  <View style="background:#FBF7F0; border:1px solid #E7DDCB; border-radius:8px; padding:14px 16px; margin-bottom:18px;">
    <Text name="text" value="$text"/>
  </View>
  <Header value="Rate each emotion's intensity (no stars = absent, 1 low, 2 medium, 3 high)"/>
  <Header value="Joy"/>
  <Rating name="joy" toName="text" maxRating="3"/>
  <Header value="Sadness"/>
  <Rating name="sadness" toName="text" maxRating="3"/>
  <Header value="Anger"/>
  <Rating name="anger" toName="text" maxRating="3"/>
  <Header value="Fear"/>
  <Rating name="fear" toName="text" maxRating="3"/>
  <Header value="Surprise"/>
  <Rating name="surprise" toName="text" maxRating="3"/>
  <Header value="Disgust"/>
  <Rating name="disgust" toName="text" maxRating="3"/>
</View>

```

The data you import is one JSON object per task, with a `data` field whose keys match the `value="$..."` references in the config:

```

{"data": {"text": "Na gode sosai, wannan labari ya faranta min rai."}}
{"data": {"text": "Ban gamsu da yadda aka gudanar da zaben ba."}}

```

After annotation, the tool exports each task with the labels attached. For the per-emotion intensity config above, an exported record carries one rating per emotion (zero meaning absent), ready to flow into the agreement script in [Data Quality](#) or into training:

```

{
  "data": {"text": "Na gode sosai, wannan labari ya faranta min rai."},
  "annotations": [
    {"completed_by": "annotator_03",
     "result": [
       {"from_name": "joy", "to_name": "text", "type": "rating", "value": {"rating": 3}},
       {"from_name": "anger", "to_name": "text", "type": "rating", "value": {"rating": 0}}
     ]}
  ]
}

```

In the tool, the task list and per-task annotation progress are managed from the data manager:

The screenshot shows the AfriAnnotate Data Manager interface. At the top, there's a navigation bar with 'AfriAnnotate' logo, a menu icon, and links to 'Projects', 'Demo · Sentiment Classi...', and 'Data Manager'. On the right, there are links for 'Dashboard', 'Data Manager' (highlighted), 'Members', 'Analytics', and 'Settings', along with a notification bell and a user profile icon labeled 'OA'.

Below the navigation bar, there's a 'Default' tab with a '+' icon. A toolbar contains 'Actions', 'Columns', 'Filters', 'Order by', and a 'Label All Tasks' button with a dropdown arrow. To the right of the toolbar are 'Import', 'Export', and a settings icon.

The main table has columns: ID, Completed, and three sentiment status columns (represented by icons: a blue square, a red circle with a slash, and a purple plus). The 'Annotated by' column is currently set to 'text' with a dropdown arrow. The table contains 12 rows of data, each representing a task with a unique ID and a text description. Row 1650 is highlighted in blue.

At the bottom right, a status bar shows 'Tasks: 12 / 12', 'Submitted annotations: 0', and 'Predictions: u'.

ID	Completed				Annotated by	text
1645	0	0	0		Loved the service, food came out warm and the waiter was attentive.	{ }
1646	0	0	0		Terrible experience — waited 40 minutes and the order was wrong.	{ }
1647	0	0	0		It was fine, nothing special but not bad either.	{ }
1648	0	0	0		The hotel was clean and the staff friendly, I'd stay again.	{ }
1649	0	0	0		Too noisy and the bed was uncomfortable. Not worth the price.	{ }
1650	0	0	0		Incredible sunset views from the balcony, kids enjoyed the pool.	{ }
1651	0	0	0		Rude reception and cold breakfast — never coming back.	{ }
1652	0	0	0		Average restaurant, decent food, slow service.	{ }
1653	0	0	0		Fantastic value for money, the tour guide was excellent.	{ }
1654	0	0	0		Taxi arrived late and the driver was grumpy.	{ }
1655	0	0	0		Service was polite but the portions were small for the	{ }

Inter-annotator agreement

Annotation agreement measures the extent to which multiple annotators assign the same labels to the same data instances. In text classification tasks, agreement is one of the most important indicators of dataset quality because it reflects the clarity of the annotation guidelines, the complexity of the task, and the consistency of the annotators. High agreement suggests that the labels are reliable and reproducible, while low agreement may indicate ambiguous definitions, insufficient annotator training, or inherently subjective phenomena.

Annotation agreement should be reported because it provides evidence that the guidelines are understandable and that the labels are reproducible. Agreement values should always be interpreted together with the task difficulty and the degree of subjectivity involved.

Why agreement matters

Annotation agreement serves various purposes:

- Evaluates the reliability of the annotated dataset.
- Identifies ambiguities in the annotation guidelines.
- Detects inconsistencies among annotators.
- Provides evidence of dataset quality for publications and benchmark releases.
- Helps determine whether a task is objectively measurable or highly subjective.

Agreement should be calculated and reported for every dataset that involves human annotation if the data is annotated by two and more annotators.

Percentage agreement

The simplest measure of agreement is percentage agreement, which calculates the proportion of instances for which annotators assigned the same label.

```
Agreement % = (Number of Agreed instances / Total Number of Instances) * 100
*For example, if two annotators label 1,000 texts and agree on 850 of them:*
Agreement % = (850 / 1000) * 100 = 85%
```

Although easy to understand, percentage agreement does not account for agreement occurring by chance and should not be the only metric reported.

Agreement between two annotators

When exactly two annotators label each instance, Cohen's Kappa is the most commonly used agreement metric. Cohen's Kappa adjusts for the amount of agreement that could occur purely by chance.

```
Kappa = (Observed Agreement - Expected Agreement) / (1 - Expected Agreement)
Or
Kappa = (Po - Pe) / (1 - Pe)
Where:
Observed agreement (Po) is the proportion of instances where the annotators actually agreed.
Po = Total number of items / Number of agreements
Expected Agreement (Pe) represents the level of agreement that would be expected to occur purely by chance, given the
distribution of labels assigned by each annotator. It is calculated by determining the probability that both annotators
independently select the same category and then summing these probabilities across all categories.
```

Cohen's Kappa is widely used in sentiment analysis, hate speech detection, topic classification, emotion classification, and many other NLP tasks involving two annotators.

```
from sklearn.metrics import cohen_kappa_score
annotator1 = [0, 1, 1, 0, 2]
annotator2 = [0, 1, 0, 0, 2]
kappa = cohen_kappa_score(annotator1, annotator2)
print(kappa)
```

Agreement among three or more annotators

Many NLP datasets use three or more annotators per instance to improve reliability and reduce the influence of individual biases.

When more than two annotators are involved, commonly used agreement measures include:

Fleiss' Kappa extends Cohen's Kappa to multiple annotators and is one of the most widely reported agreement measures in NLP datasets. It is appropriate when:

- Three or more annotators label each instance.
- Every instance receives the same number of annotations.

Fleiss kappa (k) = $(P - P_e) / (1 - P_e)$
 Where
 p is the mean of the agreement probability over all raters and
 P_e is the mean agreement probability over all raters if they were randomly assigned.

Krippendorff's Alpha is a more flexible agreement measure that:

- Supports any number of annotators.
- Handles missing annotations.
- Works with nominal, ordinal, interval, and ratio labels.
- Is increasingly recommended for modern annotation studies.

For complex annotation projects, Krippendorff's Alpha is often considered the most robust agreement metric.

Deciding the final labels

When multiple annotators label the same instance, the final label is usually determined through majority voting.

For example, in three annotators, at least two annotators must agree on a label for it to become the final label. Similarly, with five annotators, at least three annotators must agree on a label for it to become the final label. Using an odd number of annotators (3, 5, or 7) avoids ties and simplifies majority voting.

Interpreting agreement scores

Although interpretation varies slightly across fields, the following ranges are commonly used for Kappa-based agreement measures:

```
< 0.00 Poor Agreement
0.00 - 0.20 Slight Agreement
0.21 - 0.40 Fair Agreement
0.41 - 0.60 Moderate Agreement
0.61 - 0.80 Substantial Agreement
0.81 - 1.00 Almost Perfect / Excellent Agreement

*As a general guideline:*
< 0.40: dataset quality should be carefully reviewed.
0.40-0.60: acceptable for difficult or subjective tasks.
0.60-0.80: considered good agreement.
Above 0.80: considered very strong agreement.
```

For highly subjective tasks such as emotion classification, sarcasm detection, or offensiveness annotation, lower agreement scores may still be acceptable due to genuine differences in human interpretation.

What to report

When publishing a dataset, researchers should report:

1. Number of annotators.
2. Annotation procedure/guideline.
3. Final label aggregation method (e.g., majority voting).
4. Cohen's Kappa (for two annotators) or Fleiss' Kappa/Krippendorff's Alpha (for three or more annotators) agreement score.

5. Any adjudication process used to resolve disagreements.
6. Annotator-level dataset for further annotator subjectivity and disagreement research.

Transparent reporting of annotation agreement improves the credibility, reproducibility, and scientific value of the dataset.

Agreement metric guidance:

- Use percentage agreement only as a simple descriptive measure.
- Use Cohen's kappa when there are exactly two annotators.
- Use Fleiss' kappa when there are three or more annotators and each item has the same number of labels.
- Use Krippendorff's alpha when annotations may be missing or when you want a more flexible reliability measure.

Note that agreement metrics are not only those listed above. Explore more agreement metrics that suit the targeted task.

```
from sklearn.metrics import cohen_kappa_score
from statsmodels.stats.inter_rater import fleiss_kappa, aggregate_raters
import krippendorff, numpy as np

# Cohen's kappa: exactly two annotators
# Cohen's  $\kappa = (P_o - P_e) / (1 - P_e)$  where  $P_o$  = observed agreement,  $P_e = \sum_n p_{n1} \cdot p_{n2}$  (chance agreement)
cohen_kappa_score(a1, a2)

# Fleiss' kappa: items x raters matrix, equal number of raters each
# Fleiss'  $\kappa = (\bar{P} - P_e) / (1 - P_e)$  over  $n$  raters,  $k$  categories

table, _ = aggregate_raters(ratings)      #  $\rightarrow$  items x categories counts
fleiss_kappa(table)

# Krippendorff's alpha: raters x items, np.nan for missing
# Krippendorff's  $\alpha = 1 - D_o / D_e$  ( $D_o$  observed disagreement,  $D_e$  expected; handles missing data & any #raters)

krippendorff.alpha(reliability_data=data, level_of_measurement="nominal")
```



TIPS

For subjective tasks such as emotion and offensiveness annotation, lower agreement is not always a failure; it can reflect real ambiguity in human interpretation. So, lower scores can still be valid: genuine human disagreement is signal, not just noise.

Quality control

Annotation quality can be controlled before and during the annotation process using various mechanisms. The following are some of the annotation quality control methods.

Pilot annotation

Before starting large-scale annotation, dataset creators should conduct a pilot study to evaluate both the annotation guidelines and the annotators. The pilot phase helps identify unclear instructions, difficult cases, and annotators whose labeling patterns differ substantially from the rest of the group. Annotators who consistently provide random, low-quality, or highly inconsistent annotations should be identified and excluded before the main annotation process begins.

Control (gold-standard) questions

A common quality-control mechanism is to include control questions, also known as gold-standard items, whose correct labels are already known. These items are randomly inserted into the annotation workflow without informing the annotators. Annotators who repeatedly fail to label these control items correctly may be removed from the project, and their previously annotated data should be reviewed and, if necessary, excluded from the final dataset.

Determining the number of annotators

For most NLP annotation tasks, using at least three annotators per instance is a common practice for ensuring annotation quality. An odd number of annotators (e.g., 3, 5, or 7) enables majority voting to determine the final label. In general, increasing the number of annotators per instance improves the reliability and robustness of the dataset by reducing the impact of individual biases. However, annotation cost and annotator availability often limit the number of annotators that can be employed.

When human resources are limited, annotation can be performed by two annotators. In such cases, dataset creators may either retain only the instances on which both annotators agree or introduce an adjudication process, where disagreements are resolved through discussion or by an expert annotator who makes the final decision.

 [Cite this page](#)  13 min read

CONTRIBUTORS  (<https://github.com/abumafrim>)  (<https://github.com/shmuhammadd>)

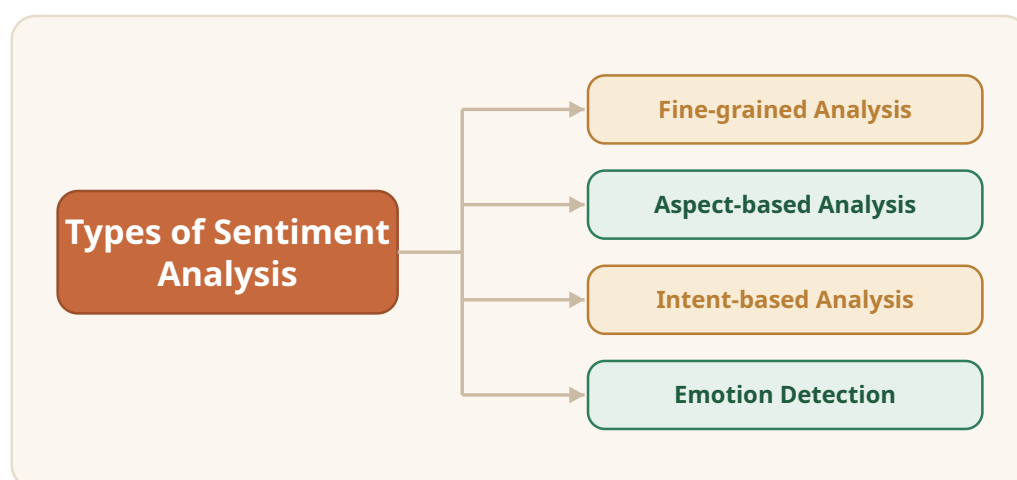
Sentiment Analysis

Sentiment analysis in this playbook refers to labeling text according to the polarity of the expressed attitude toward a subject, product, event, or experience. The most common labels are positive, negative, neutral, and mixed. Depending on the project, sentiment may be annotated at the document level, sentence level, or aspect level.

The Sentiment Analysis Annotation Guidelines Template is a structured framework designed to standardize the process of annotating textual data for sentiment analysis. This template is particularly crucial in industries like customer service, marketing, and social media analytics, where understanding the sentiment behind user-generated content is essential. By providing clear instructions and examples, the template ensures that annotators can consistently label data as positive, negative, or neutral. For instance, in a customer feedback analysis project, this template helps teams align on how to interpret ambiguous phrases, ensuring the dataset's reliability and accuracy. The importance of such a template cannot be overstated, as it directly impacts the quality of machine learning models trained on the annotated data.

Types of sentiment analysis

Sentiment analysis can be annotated into one or more of the following annotation types.



Fine-grained (Scaling) analysis: Graded sentiment analysis assigns scores on a scale, providing a more nuanced view of sentiment intensity. This approach helps gauge the strength of emotions expressed in the text. For example, a review might be labeled as "very positive," "slightly positive," "neutral," "slightly negative," or "very negative."

Aspect-based analysis: This focuses on identifying sentiment towards specific aspects or features of a product, service, or topic. For instance, in a hotel review, aspect-based analysis might determine positive sentiment towards the location but negative sentiment towards the cleanliness. For a smartphone review, it separately analyzes battery, screen, camera and performance to understand customer sentiment for each aspect.

Intent-based analysis

This analysis type can detect the motive behind a text, whether the author is trying to express an opinion, make a recommendation, ask a question, or express a need. Understanding intention is important in customer service, market research, and targeted advertising. For example, a customer tweets, 'I wish Company X's product had a longer battery life.' This indicates dissatisfaction and a desire for improvement (intent to recommend a feature change). This helps Company X handle the negativity and use this feedback to improve their products.

Sentiment analysis labels definition

While the types of sentiment targets can differ, the annotation labels can be two (positive and negative), three (positive, negative, and neutral), or all of the labels below.

- **Positive:** a message that conveys a clearly favorable attitude, such as satisfaction, approval, praise, or happiness toward a subject, product, or experience.
- **Negative:** a message that conveys a clearly unfavorable attitude, such as dissatisfaction, disapproval, criticism, or frustration toward a subject, product, or experience.
- **Mixed:** a message that includes both positive and negative sentiments, either about different aspects of the same subject or within the same statement.
- **Neutral:** a message that expresses no clear positive or negative sentiment, typically presenting factual, descriptive, or objective information without emotional judgment.

Examples of sentiment analysis are as follows:

Label	Decision rule	Example
Positive	Overall attitude is favorable, even with minor complaints.	"I love this phone, best I've owned."
Negative	Overall attitude is unfavorable.	"This app keeps crashing, useless."
Mixed	Use only when both are clearly present and roughly balanced.	"Great camera but the battery dies fast."
Neutral	Default when no evaluative stance is expressed.	"The package arrived on Tuesday."

Labeling a sentiment in the AfriAnnotate editor:

AfriAnnotate

Demo · Sentiment Classi...

Project sections

AA

Default

+

Actions

Columns

Filters

Order by

Label All Tasks

ID

Completed

text

str

1647

It was fine, nothing special but not bad either.

1648

The hotel was clean and the staff friendly, I'd stay again.

1649

Too noisy and the bed was uncomfortable. Not worth the price.

1650

Incredible sunset views from the balcony, kids enjoyed the pool.

1651

Rude reception and cold breakfast — never coming back.

1652

Average restaurant, decent food, slow service.

1653

Fantastic value for money, the tour guide was excellent.

1654

Taxi arrived late and the driver was grumpy.

Tasks: 10 / 12

Submitted annotations: 0

Predictions: u

[Cite this page](#) 3 min read

CONTRIBUTOR  (<https://github.com/abumafrim>)

Page 142 of 150

Emotion Analysis

What is Emotion Analysis?

Emotion detection or emotion classification is a task that aims to identify and classify the emotions expressed in a piece of text. Unlike sentiment analysis, which usually categorizes text as positive, negative, or neutral, emotion analysis seeks to recognize more specific emotional states such as anger, joy, sadness, fear, surprise, disgust, and neutral.

Emotion annotation goes a step further than sentiment analysis by analyzing specific emotions and classifying the text into categories such as joy, anger, sadness, fear, or surprise. By recognizing these, companies can better understand customer responses, which helps them respond to specific issues accordingly. For example, if you can identify any frustration in a customer complaint, you can address the issue immediately and prevent escalation. Emotion can be annotated into one of the following approaches.

Single-label Emotion Analysis

In single-label annotation, a text has either no emotion or only one of the emotions for the given list of emotions; each text is assigned one dominant emotion.

Id	Text	Emotion
sample_01	Never saw him again.	Sadness
sample_02	I love telling this story.	Joy
sample_03	How stupid of him.	Anger
sample_04	None of us did.	Neutral
sample_05	I can't believe it! I won the scholarship! This is amazing!	Joy

Multi-label Emotion Analysis

In multi-label emotion analysis, a text may express no emotion, a single emotion, two emotions, or multiple emotions simultaneously. A sample multi-label emotion annotation interface is shown below. The use of checkboxes allows annotators to select one or more emotion categories, along with the corresponding intensity level for each selected emotion. In multi-label emotion annotation, recording emotion intensities is common practice because different emotions may be expressed with varying degrees of strength within the same text. Therefore, intensity annotations provide additional information beyond the mere presence or absence of an emotion.

AfriAnnotate Projects / Demo · Emotion (multi-L... / Labeling Dashboard

They cancelled the festival again, what a waste.

Rate the intensity of each emotion — no stars = not present, 1 = low, 2 = medium, 3 = high

Anger
☆☆☆

Disgust
☆☆☆

Fear
☆☆☆

Sadness
☆☆☆

Joy
☆☆☆

Surprise
☆☆☆

Regions Info Rel His Co
Manual By Time

Labeled regions will appear here
Start labeling and track your results using this panel
[Learn more](#)

#2440 1 of 1 < > ↶ ↷ ⌂

Skip Submit

Given annotations from multiple annotators, the final emotion labels can be determined through an aggregation process. When only binary emotion labels are available, a majority-vote approach can be used to determine the final label for each emotion category. However, when emotion intensity ratings are also collected, both the emotion labels and their intensities can be considered when making the final decision.

The following rule can be used to determine the final emotion label based on ratings from multiple annotators. The final label is binary, where 1 indicates the presence of the emotion, and 0 indicates the absence of a significant emotional expression.

For datasets annotated without considering intensities, decisions can be made by a simple majority vote. For emotion with intensity annotation, the following decision rule can be applied:

The majority vote works for simple cases. However, It may not work if you have an even number of annotators. For example, given four annotator scores (0,0,3,3): 0 (no-anger), 3 (high-anger), there is no majority vote in this case. IF (at least two annotators assign a non-zero intensity score, e.g., 1, 2, or 3) AND (the average intensity score exceeds a predefined threshold), THEN the final emotion label = 1. ELSE the final emotion label = 0.

This approach combines annotator agreement and emotion intensity, resulting in a more reliable representation of the emotional content of the text.

Conditions for Determining the Final Emotion Label

1. At least 2 people must select 1(low), 2(medium), or 3(high): This ensures that there is a basic level of agreement among the annotators that the content contains some level of emotion (low, medium, or high).

2. Avg score > threshold: The average score given by all annotators must be greater than a predefined threshold. This ensures that the intensity of the emotion is significant enough to be considered present.

Since we have a scale from 0 to 3, a threshold of 0.5 can be a good choice. The below are example scenarios.

Example 1

- Annotator scores: 0, 0, 3,3
- Average score: $(0 + 0 + 3 + 3) / 4 = 1.5$
- Result: Since $1.5 > 0.5$, final label = 1 (emotion present).

Example 2

- Annotator scores: 1, 2, 1
- Average score: $(1 + 2 + 1) / 3 = 1.33$
- Result: Since $1.33 > 0.5$, final label = 1 (emotion present).

Example 3

- Annotator scores: 1, 1, 0
- Average score: $(1 + 1 + 0) / 3 = 0.666$
- Result: Since $0.66 > 0.5$, final label = 1 (emotion present).

Example 4

- Annotator scores: 1, 1, 0, 0,0
- Average score: $(1 + 1 + 0 + 0 + 0) / 5 = 0.4$
- Result: Since $0.4 < 0.5$, the final label = 0 (no emotion present).

Example 5

- Annotator scores: 1, 2, 3, 2
- Average score = $(1 + 2 + 3 + 2) / 4 = 8 / 4 = 2.0$
- Result: Since $2.0 > 1.5$, final label = 1 (emotion present).

This method combines both agreement among annotators and the intensity of the emotion, providing a balanced evaluation. Majority vote does not consider the intensity and may miss out on subtle emotional content. A sample format of the multi-label emotion dataset is shown below. The final binary emotion labels are then derived using the aggregation procedure described above.

Id	Text	Anger	Fear	Joy	Sadness	Surprise
sample_01	Never saw him again.	0	0	0	1	0
sample_02	I love telling this story.	0	0	1	0	0
sample_03	How stupid of him.	1	0	0	0	0
sample_04	None of us did.	0	0	0	0	0
sample_05	I can't believe it! I won the scholarship! This is amazing!	0	0	1	0	1

Emotion Intensity

Annotators indicate the emotions that are likely conveyed in the text and their intensity levels, i.e., low (1), medium (2) or high (3). The scores that are associated to the different intensity levels (0 [no emotion] to 3 [high emotion]) are then averaged. This average score is used to classify the emotion intensity based on its proximity to the predefined values corresponding to the intensity (i.e., low (0), 1 (low), 2 (medium), and 3 (high)).

The following is how to make majority vote during intensity annotation if the intensity labels are in likert scale 0 - 3 (0 is - no emotion, 1 - low intensity, 2 - medium intensity, and 3- high intensity).

Example 1: Give an example of an emotion and a post [e.g., angry].

- Annotator scores = [1, 2, 2, 3]
- Average score = $(1 + 2 + 2 + 3) / 4 = 2.0$
- Intensity class = 2 (Moderate amount of emotion)

Example 2:

- Annotator scores = [0, 0, 1, 1]
- Average score = $(0 + 0 + 1 + 1) / 4 = 0.5$
- Intensity class = 1 (Low amount of emotion)

Example 3:

- Annotator scores = [3, 3, 3, 2]
- Average score = $(3 + 3 + 3 + 2) / 4 = 2.75$
- Intensity class = 3 (High amount of emotion)

Example 4:

- Annotator scores = [0, 0, 0, 1]
- Average score = $(0 + 0 + 0 + 1) / 4 = 0.25$
- Intensity class = 0 (No emotion)

A sample format of the multi-label emotion intensities dataset is shown below. Each emotion is assigned an intensity score ranging from 0 (not present) to 3 (high intensity).

Id	Text	Anger	Fear	Joy	Sadness	Surprise
sample_01	Never saw him again.	0	0	0	2	0
sample_02	I love telling this story.	0	0	2	0	0
sample_03	How stupid of him.	2	0	0	0	0
sample_04	None of us did.	0	0	0	0	0
sample_05	I can't believe it! I won the scholarship! This is amazing!	0	0	3	0	3

 [Cite this page](#)  6 min read

CONTRIBUTOR  (<https://github.com/abumafrim>)

Hate Speech Analysis

What is Hate Speech Analysis?

Hate speech detection is a task that focuses on identifying text that contains hate, offensive, abusive, or discriminatory language directed at individuals or groups based on attributes such as race, ethnicity, religion, gender, nationality, disability or other identity factors. The goal of this task is to automatically classify whether a piece of text expresses hate, hostility, or incitement to harm, often differentiate it from non-hateful content such as criticism, disagreement, or neutral statements. Hate speech detection is widely used in content moderation systems among various social media platforms to reduce online harassment and promote a safer digital communication ecosystem.

Researchers used various class labels, such as Muhammad et al. (2025), who used hate, abusive, normal and indeterminate as class labels, while Ayele et al. (2022, 2023) utilized hate, offensive, normal, and unsure class labels. Ayele et al. (2024) used class labels such as hate, offensive, normal and indeterminate. Davidson et al. (2017) used hate, offensive and neither class labels, while Davidson et al. (2019) used hate, and offensive class labels. Mathew et al. (2021) used hate, offensive, normal and undecided class labels.

The most agreed-upon definitions of the hate speech classes are as follows:

- **Offensive speech:** is any form of bad language expressions including rude, impolite, insulting, or belittling utterances intended to offend or harm an individual.
- **Hate speech:** is language content that expresses hatred towards a particular group or individual based on their group identities such as race, ethnicity, religion, gender, disability, political affiliation, or other characteristics. It also includes threats of violence associating group identities.
- ****Normal**** is any form of expression that does not contain any bad language belonging to any of the above classifications.
- ****Indeterminate**** is any tweet that is not readable or is completely written in a language other than your language of annotation.

Hate speech can target one or many targets during expression. Ayele et al (2024) presented a more general list of hate speech target labels as follows:

- **Ethnic:** if the hatred tweet targets an ethnic group identity
- **Religion:** if the hatred tweet targets a religious group identity
- **Gender:** if the hatred tweet targets a particular gender group identity
- **Disability:** if the hatred tweet targets disabled group identity
- **Politics:** if the hatred tweet targets people/entities due to political ideology
- **Unidentified target:** if the hatred tweet's target is not clearly identified/known.
- **Other:** if the hatred tweet targets other groups/group identities such as sexual orientation, racism etc.

Some studies, such as Ayele et al (2024) also examine and rate the severity of hate and offensive messages with rating scales from 1-5.

AfriAnnotate

Projects / Demo · Hate Speech (cat... / Labeling

Dashboard

AA

Thank you all for the warm welcome to the community.

What is the text category?

☐ Offensive^[1]

☒ Hate^[2]

☐ Normal^[3]

☐ Indeterminate^[4]

How hateful is this text?

Who is the target of the hate?

☒ Ethnicity^[6]

☐ Religion^[7]

☐ Disability^[8]

☐ Gender^[9]

☒ Politics^[0]

#2442
1 of 1

<

>

↶

↷

↺

↻

Skip

Submit

▼

Regions

Info

Rel

His

Co

Manual

By Time

⌵

⊞

Labeled regions will appear here

Start labeling and track your results using this panel

[Learn more](#)

 [Cite this page](#) ⌚ 2 min read

CONTRIBUTOR  (<https://github.com/abumafirim>)

Page 149 of 150

Text Generation

In development

This chapter is being written. It'll appear here once the authors are happy with it. Nothing to see yet. Check the sidebar for chapters that are ready, or come back later.